Universidad Carlos III
Departamento de Tecnología Electrónica
Proyecto Fin de Carrera

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
ÁREA DE TECNOLOGÍA ELECTRÓNICA

SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS

AUTOR: **RAQUEL RUIZ CAÑAMERO**
DIRECTOR: **JUAN VÁZQUEZ MARTÍNEZ**
FECHA: **SEPTIEMBRE 2015**

“Miro hacia atrás y me pregunto… ¿quién soy?
Soy un poquito de cada uno de vosotros. Soy mi padre y mi madre, mis hermanos y sobrinos. Soy mi marido y mi hijo, pero sobre todo, soy un poco de cada persona que he amado, que me ha enseñado a querer y a vivir.”

A mi gran familia.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ACRÓNIMOS

ACK ..... "Acknowledgement" ó Acuse de recibo.
CPU ..... "Central Processing Unit"
CTS ..... "Clear To Send" ó Preparado para enviar
DGT ..... Dirección General de Tráfico
DSP ..... Digital Signal Processor
GSM ..... Global System for Mobile
IR ..... Infrarrojos
LD ..... Laser Diode
LDR ..... Light Dependent Resistence
LED ..... Light Emiting Diode
PAN ..... Red de Área Local Privada
PC ..... Personal Computer
PIC ..... Controladores de Interfaz Periférico
PWM ..... Pulse-Width Modulation
RF ..... Radiofrecuencia
RGB ..... "Red, Green, Blue" para indicar la composición del color
RTS ..... "Request to Send" o Solicitud de envío
Rx/Tx ..... Reception / Transmision [Comunicaciones Puerto Serie]
SO ..... Sistema Operativo
UART ..... Universal Asynchronous Receiver-Transmitter
UE ..... Unión Europea
WPAN ..... Wide Personal Area Network

RESUMEN

En este proyecto se presenta el desarrollo de un sistema de iluminación inalámbrico para ciclistas mediante la integración de elementos existentes en el mercado actual.

Para ello se ha realizado un estudio de la necesidad, la elección de componentes para el posterior diseño e integración del sistema, y la programación de las rutinas para que se ejecute el comportamiento deseado.

Indicar que se definen dos módulos principales, que en el proyecto llamaremos NODO (recoge las señales de los actuadores) y COORDINADOR (recibe la información y tras interpretarla procede a la señalización lumínica).

El **NODO** tiene una doble función. Por un lado recoger las señales de los actuadores, procesarlas y enviarlas al COORDINADOR. Por el otro, monitorizar el estado del sistema en el módulo remoto.

El **COORDINADOR** tiene como misión fundamental encender el panel de señalización led y enviar el feedback correspondiente al emisor.

Para la comunicación se utilizarán módulos Xbee, y se configurarán en modo “Conexión Transparente”.

Como sistema de control, utilizaremos placas Arduino UNO. Se ha utilizado Processing (similar a C++) como lenguaje de programación.

Los actuadores que funcionan como elementos de entrada son pulsadores.

ESQUEMA GENERAL

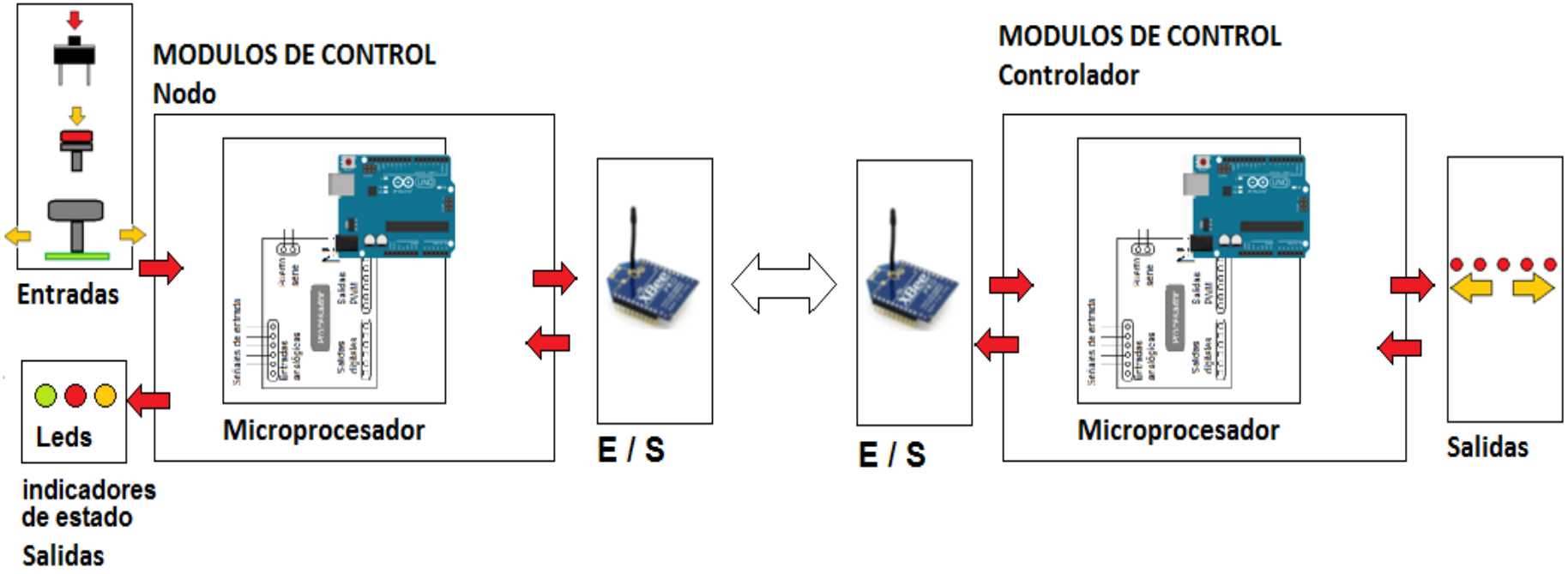


Ilustración 1. Resumen - Esquema general

1 MOTIVACIÓN Y OBJETIVOS

El objetivo principal del presente proyecto es diseñar un sistema de iluminación ciclista mediante **la integración de diversos módulos y elementos existentes** actualmente en el mercado.

En los últimos años se ha producido un incremento de usuarios de bicicleta debido a que es un medio de transporte más económico y respetuoso con el medio ambiente que los vehículos a motor, resolviendo problemas de congestión en áreas saturadas de tráfico rodado y además porque su uso regular constituye un hábito saludable.

Por estas razones, cada vez más países han definido políticas para promover el uso de la bicicleta.

Aunque la bicicleta tiene sus ventajas, hay que admitir que su utilización entraña indudablemente altos riesgos derivados del uso compartido de las vías de circulación con otros vehículos.

Para la utilización compartida de dichas vías existen unas normativas tanto a nivel nacional como europeo que tipifican la **obligatoriedad de la señalización**, bien mediante gestos que realiza el usuario, bien mediante elementos externos, generalmente sonoros o lumínicos, que permiten al resto de vehículos prever las maniobras que se realizan en los desplazamientos.

Si bien para aquellos vehículos propulsados a motor existen elementos externos lumínicos integrados en los propios vehículos, regulados y obligatorios, que facilitan dicha señalización, en lo referente a los ciclos sólo existe normativa referente a la iluminación de posicionamiento (luces delantera y trasera, así como reflectantes y catadióptricos), siendo necesaria la realización de gestos para indicar los cambios de sentido, dirección e incluso parada.

Para realizar estos gestos y señales, el usuario debe soltar el manillar con una de sus manos, lo que implica una desestabilización en momentos en los que su atención por motivos de seguridad debe estar focalizada en la maniobra a realizar.

Con el objeto de **mejorar la seguridad del usuario** así como la comodidad y facilidad a la hora de realizar maniobras, promoviendo la utilización de la bicicleta, se estudiará la integración de distintos elementos para la obtención de un producto final consistente en un sistema de **señalización lumínica** portátil adaptado a ciclistas.

1.1 ESTRUCURA Y ALCANCE DE LA MEMORIA

El alcance del presente proyecto, de cara a la consecución de los objetivos marcados, es **la integración de dispositivos** existentes en el mercado para la consecución de un sistema de intermitencia inalámbrico consistente en **dos módulos (emisor y receptor)**.

Para ello, dividiremos el proyecto en las siguientes fases:

- Introducción: Podemos dividirla en dos grandes bloques. Por un lado un breve estudio sobre las normas vigentes de señalización de las maniobras por parte de los ciclistas. Posteriormente se presenta la idea inicial de producto y su evolución en el tiempo.

- Análisis de la situación actual: Contiene un análisis de la situación y necesidad actual, así como de los dispositivos similares existentes en el mercado.

- Diseño del sistema de señalización: Análisis y elección de los elementos que conformarán nuestro producto.

- Memoria del sistema de señalización: Una vez definidos los componentes, y teniendo en cuenta las restricciones del producto, desarrollamos la memoria estructurando su contenido en función de si estamos trabajando la parte de integración Hardware o de desarrollo de programas Software.

  Al finalizar este apartado, disponemos de las configuraciones, rutinas y programas (módulos de control y transmisión), y de las características y configuración Hardware.

- Pruebas y validación: Descripción de las pruebas realizadas en el desarrollo del proyecto.

- Presupuesto: Valoración económica de los costes totales estimados para la fase de desarrollo equivalente a la realización del proyecto.

- Conclusiones y trabajos futuros: Gracias a la modularidad y aplicación directa en el mercado, así como a la necesidad de soluciones que mejoren la experiencia del usuario final, durante la realización del presente proyecto se han ido incorporando ideas sobre posibles mejoras o nuevas funcionalidades que aportan valor añadido. En este apartado se recogerán y analizarán brevemente, para extraer conclusiones y presentar mejoras futuras.

Por último, se incluyen en el proyecto los apartados correspondientes a los índices de Tablas e Ilustraciones, Bibliografía y Anexos, destacando los referentes a los PLANOS y PROGRAMAS ya que muestran el Hardware integrado y el software desarrollado.

2 INTRODUCCIÓN

"Ir en bicicleta es más sano que conducir, y reduce el ruido, la contaminación y los problemas de congestión. Medidas de seguridad sencillas como llevar un casco y utilizar vías ciclistas, puede reducir el riesgo de daños en caso de accidente"

Unión Europea - web

Bien sea por motivos económicos, ecológicos o relacionados con la salud, actualmente se está viviendo un repunte en la utilización de la bicicleta como medio habitual de transporte, fundamentalmente en las ciudades que están haciendo grandes esfuerzos por adaptar las vías a las nuevas necesidades sociales aplicando medidas como la construcción de carriles bici, o la puesta a disposición de los usuarios de plataformas de alquiler y aparcamiento de bicicletas.

Sin embargo, no hay que olvidar que la circulación por la vía pública entraña una serie de responsabilidades derivadas del uso compartido de la misma con otros usuarios.

Este hecho hace necesario para el usuario no solo conocer sino respetar el conjunto de normas establecidas.

Este conjunto de normas unido a las necesidades del mercado afectan al diseño de nuestro producto.

Por otro lado, a continuación se incluye una descripción de lo que se considera ciclo y bicicleta, ya que conociendo el vehículo al que va dirigido el producto es útil para entender la necesidad del mismo:

Ciclo[1]:

"Un ciclo es un vehículo con al menos dos ruedas propulsado por la energía de la persona que lo conduce mediante pedales o manivelas y que debe tener un ***sistema de frenado eficiente****, equipado con* ***un timbre*** *que se escuche a una distancia suficiente (sin llevar ningún otro sistema de alarma sonoro) y equipado con* ***reflectantes rojos en la parte trasera*** *así como dispositivos que aseguren que la bicicleta disponga de una* ***luz blanca o amarilla delantera y una luz roja trasera."***

Bicicleta:

"*Ciclo de 2 ruedas.*[2] "

Con la reciente entrada en vigor del nuevo reglamento de vehículos, se incluye la nueva categoría de "**Bicicleta de pedaleo asistido**" que engloba a aquellas bicicletas con potencia no superior a 0.5 kW cuyo motor se detiene bien al cesar el pedaleo del conductor, bien al superar una velocidad de 25 km/hora.

[1] Descripción acordada en la **Convención de Viena de 1968**. En dicha convención se unifican muchos aspectos relativos a la circulación en Europa, como el formato de las señales de tráfico.
[2] http://www.dgt.es

2.1 LA SEÑALIZACIÓN EN LA SEGURIDAD VIAL

"Un simple gesto, empujar la palanca hacia arriba o hacia abajo, y todos los conductores alrededor saben si usted va a girar, a cambiar de carril o a estacionar y pueden actuar en consecuencia. Conducir bien también es maniobrar sin sorpresas."

Carlos NICOLÁS FRAILE - Dgt.es

La formación y concienciación de los usuarios, la obligatoriedad de elementos como el casco y luces de posición, y la adecuación de las vías, si bien resultan de utilidad no son suficientes para mejorar significativamente la seguridad tanto de los ciclistas como del resto de usuarios de las vías.

A pesar de la importancia constatada de **anunciar previamente las maniobras de una forma sencilla y visible**, y de que el resto de vehículos tienen una normativa específica y marcada sobre los sistemas de intermitencia, aún no se dispone de elementos integrados en las bicicletas para poder realizar dicha señalización de una forma eficaz, a pesar de que los usuarios de estos vehículos son especialmente vulnerables en caso de choque.

Esta falta de elementos, se debe en cierta medida a que no existe regulación sobre la señalización lumínica de las maniobras inherentes a la conducción (giro, emergencia, frenadas...) **Contemplándose únicamente señalización mediante signos que implican directamente un menor control del vehículo**.

2.1.1 NORMATIVA APLICABLE (UE - ESPAÑA)

Actualmente no se dispone de normativa común a nivel nacional / europeo acerca del uso de intermitentes en ciclos de dos ruedas, si bien es cierto que se establece la obligatoriedad de señalizar los giros mediante movimientos específicos de los brazos que implican soltar el manillar y por tanto un menor grado de control sobre el propio vehículo.

Sin embargo, si nos centramos en la normativa aplicable a otros vehículos similares a motor como pueden ser los ciclomotores, encontramos algunas indicaciones referentes a color y posición que podemos incorporar a nuestro producto con objeto de homogeneizar las señales para facilitar su interpretación por los diferentes usuarios.

Normativa UE: El lugar de referencia es la página de la Comisión Europea[3] en donde se estructura la información disponible en torno a los siguientes puntos:

- Reglamento sobre vehículos.
- Reglamento sobre el uso de cascos.
- Normas de circulación.

A nivel europeo, nos encontramos con normas generalistas así como manuales de buenas prácticas y estudios para la mejora de la seguridad y el fomento del uso de la bicicleta.

No se dispone de normativa común para el uso de intermitencias en ciclos.

[3] http://ec.europa.eu/transport/road_safety/specialist/knowledge/pedestrians/

Normativa Española: El lugar de referencia es la página de la Dirección General de Tráfico[4], que proporciona normativa, estudios y referencias para la obtención de las distintas normas a nivel nacional.

Las reglas generales respecto alumbrado de vehículos aparecen recogidas en el artículo 15 del Real Decreto 2822/98 de 23 de Diciembre por el que se aprueba el Reglamento General de Vehículos, que entre otros, recoge:

Obligatorio para todos los vehículos:

- Luces y dispositivos reflectantes que siendo dobles tengan la misma finalidad: se colocarán simétricamente a la misma distancia de los bordes del vehículo, y se corresponderán en color e intensidad.

Este punto es importante a la hora de realizar nuestro diseño, por lo tanto los intermitentes deben estar situados de forma simétrica respecto al eje central y corresponderse en color e intensidad.

2.1.2 MANIOBRAS

Definimos maniobra como cualquier variación en la situación o posición de un vehículo en la calzada. Su realización puede afectar al resto de usuarios, por lo que es necesario señalizarlas y para el caso de los ciclistas está reglada por normativa comunitaria.

Para la señalización de los giros en ciclos, existe abundante documentación relativa al empleo de señales manuales, que como ya hemos comentado llevan implícita una pérdida de control sobre la conducción al tener que separar necesariamente las manos del volante.

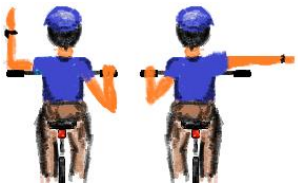
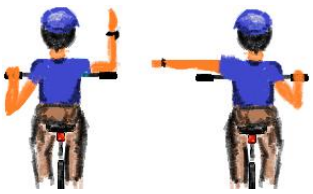
	Giro a la derecha	**Giro a la izquierda**	**Parada o freno**
Señal 1	**Brazo Izquierdo:** doblado hacia arriba con la palma de la mano extendida	**Brazo izquierdo** horizontal con la palma de la mano extendida hacia abajo	Movimientos cortos y rápidos del brazo repetidas veces en sentido: arriba / abajo.
Señal 2	**Brazo derecho:** Horizontal con palma de la mano extendida hacia abajo	**Brazo derecho** doblado hacia arriba con la palma de la mano extendida	
			

Tabla 1. Señalización de Maniobras

[4] www.dgt.es

2.2 PLANTEAMIENTO INICIAL DEL PRODUCTO

"La mente es como una hoja en blanco sobre la que la experiencia va grabando sus propios caracteres…

… Podemos distinguir dos tipos de experiencia. Una experiencia "externa", que nos afecta por vía de la sensación, y una experiencia "interna", que lo hace mediante la reflexión.
La sensación y la reflexión son, pues, las dos formas de experiencia de las que derivan todas nuestras ideas.

John Locke (1632 – 1704)

En 2006 me puse en contacto con una empresa que me remitió una muestra de una botonera diseñada para integrarse en mochilas o prendas textiles y controlar iPad.

Su concepto y funcionamiento basado en hilos conductores, flexibles, resistentes a la humedad y que podían tejerse era sencillo, y a la vez abría un amplio abanico a nuevos proyectos y desarrollos.

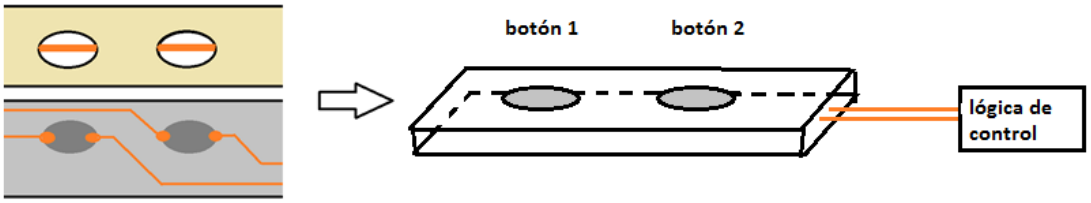


Ilustración 2. Botonera textil.

Aunque hoy en día es relativamente sencillo adquirir hilo conductor a través de internet, en aquel momento encontré que prácticamente sólo lo distribuían centros tecnológicos textiles o empresas muy especializadas.

Gracias a contactos personales con un Centro Tecnológico de Confección recibí una bobina de dicho hilo e inmediatamente comencé a hacer pruebas.

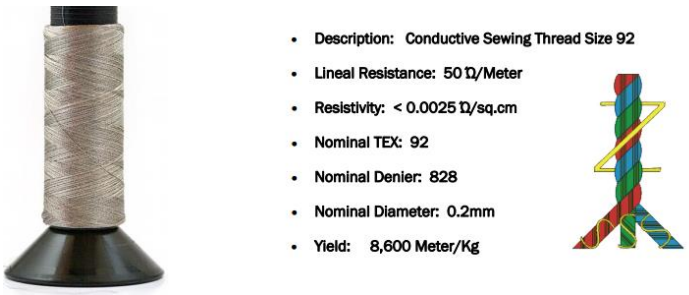


Ilustración 3. Hilo conductor[5]

[5] Fuente: https://www.sparkfun.com/datasheets/E-Textiles/260151023534oz.pdf

Es en este momento cuando surgió la idea de crear una mochila con botoneras textiles que dispusiera de indicadores luminosos para la señalización de los giros ciclistas.

IDEA INICIAL: Sin control desde el manillar.

Sistema de señalización lumínica para ciclistas utilizando hilo conductor, pulsadores textiles y módulos de control, alimentación e iluminación.

- *Mochila, chaleco, panel…*

MOCHILA:

Funcionamiento básico: asas de la mochila con pulsadores, cuyas señales se transmiten mediante hilo conductor a un panel luminoso en la parte trasera del dispositivo.

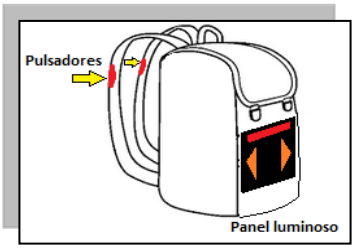


Ilustración 4, Mochila

Pros:

- Pulsación sencilla: mediante un contacto único en el asa del dispositivo.
- Alta visibilidad: al tener disponible para utilizar toda la parte trasera correspondiente a la espalda del usuario.
- Cumplimiento de normativa: luces ubicadas simétricamente respecto al eje.
- Diseño sencillo y económico: Al no precisar de dos elementos diferenciados (emisor / receptor separados físicamente) se eliminan problemas de interferencias, se reduce el consumo de baterías y de componentes necesarios para la fabricación.
- Sistema take away: El usuario se lleva consigo la mochila una vez baja del vehículo, con lo que se reduce el riesgo asociado a robos.

Contras:

- Falta de control del usuario: Debido al diseño, el ciclista no tiene un control ni lumínico ni sonoro sobre si ha pulsado correctamente, o si ha procedido a apagar la señal una vez realizada la maniobra. Esto puede generar confusión y a la larga ser contraproducente.
- Desarrollos e integraciones futuras: Al no disponer de elementos de comunicaciones inalámbricos, no se hace posible incorporar futuros módulos para la detección del frenado de una forma distinta al uso de acelerómetros.

CHALECO:

Funcionamiento básico: Muy similar al de la mochila.

Como ventaja principal encontramos el aumento de visibilidad del usuario.

Por el contrario, es más sensible al peso de las baterías, y una vez finalizado el trayecto es menos cómodo para el usuario, ya que debe disponer de algún lugar para guardarlo.

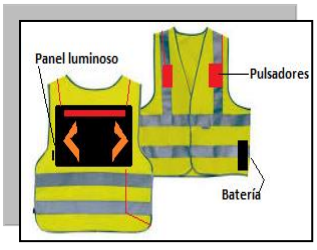


Ilustración 5. Chaleco

PANEL FLEXIBLE:

Funcionamiento básico: Equivalente a Mochila y Chaleco consta de un panel luminoso y de unas cinchas ajustables para su colocación.

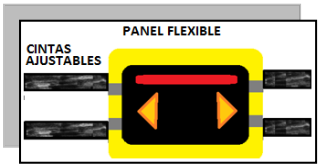


Ilustración 6. Panel

EVOLUCIÓN CONTROL EN EL MANILLAR.

La problemática detectada en el control del sistema se resuelve si se incluye un elemento transmisor en el manillar que envíe las señales (de forma inalámbrica) al receptor.

- *Emisor fijo (manillar) o móvil (caja externa con cuadro de mando)*
- *Receptor en mochila, chaleco o panel flexible.*

Evaluando la forma de poder solventar el problema del control del usuario sobre el propio sistema, se optó por dividir el producto en dos partes físicas diferenciadas:

- **Emisor** con pulsadores en el manillar
- **Receptor** con tratamiento de la señal recibida que transformase la información en señales luminosas visibles.

Para la parte receptora se podía emplear cualquiera de las propuestas anteriores, e incluso combinarlas en función de las preferencias del usuario por lo que en este punto los esfuerzos en cuanto al diseño se centraron en el emisor, para el cual hubo dos ideas diferentes respecto a líneas de trabajo:

- Emisor fijo en el vehículo: Manillar.
- Emisor móvil: caja con cuadro de mandos.

MANILLAR:

Esta opción se basa en que el manillar de la bicicleta es generalmente un cilindro hueco sobre el que se colocan unas manetas de materiales plásticos, que son las que el usuario sujeta durante la conducción del vehículo.

Si se aprovecha la cavidad para que albergue la electrónica necesaria, y en la parte externa se sitúan los pulsadores y huecos para que se realice la transmisión de la señal de una forma satisfactoria, aportamos una solución integrada y funcional que resuelve la problemática anteriormente presentada. Nos queda sin embargo el problema de que el espacio disponible es muy limitado.

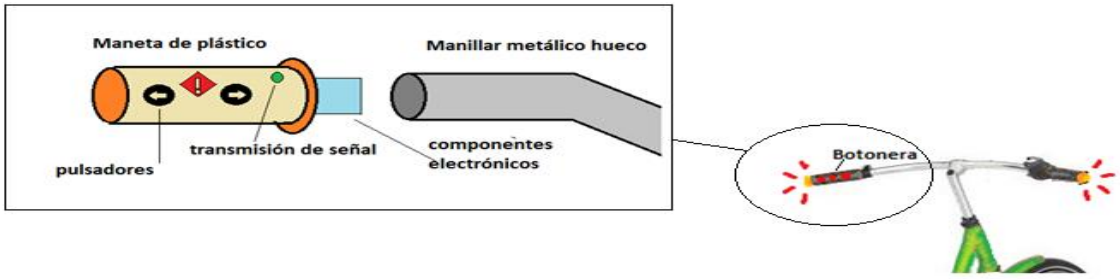


Ilustración 7. Manillar

¿Cómo solucionar el problema del espacio? Si se inyectasen los circuitos sobre superficies flexibles podrían adaptarse al hueco curvo disponible, pero esta solución requeriría de un desarrollo específico objeto en sí mismo de otro proyecto.

CAJA CON CUADRO DE MANDOS:

Se trata de la opción menos restrictiva respecto a espacio y tecnologías, dado que se fabricaría ad hoc en función de la electrónica y las necesidades finales.

Incorpora en la parte trasera una pinza para anclaje al manillar.

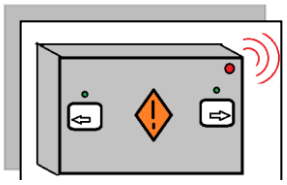


Ilustración 8. Cuadro de mandos

BUSQUEDA DE UNA SOLUCIÓN UNIFICADA DE INTEGRACIÓN

Dadas las posibilidades de adaptación y ampliación en función de las necesidades de los usuarios, se decide enfocar el trabajo a la realización de un proyecto de integración.

Teniendo en cuenta la idea inicial respecto a funcionalidades, las demandas del mercado y la normativa existente, se establece como objetivo un sistema de luces intermitentes:

- ***Sistema "vivo", escalable:*** *para poder integrar futuras aplicaciones y adaptarse a nuevas normativas.*
- ***Control desde el manillar (emisor)***
- ***Uno o más receptores****.*
- ***Comunicación inalámbrica****.*
- ***Sistema seguro****: poco sensible a interferencias del entorno.*
- *Los materiales empleados deben* ***seguir la norma*** *en caso de existir.*
- *Sistema "Eco-friendly": baterías recargables, sistema universal de carga.*

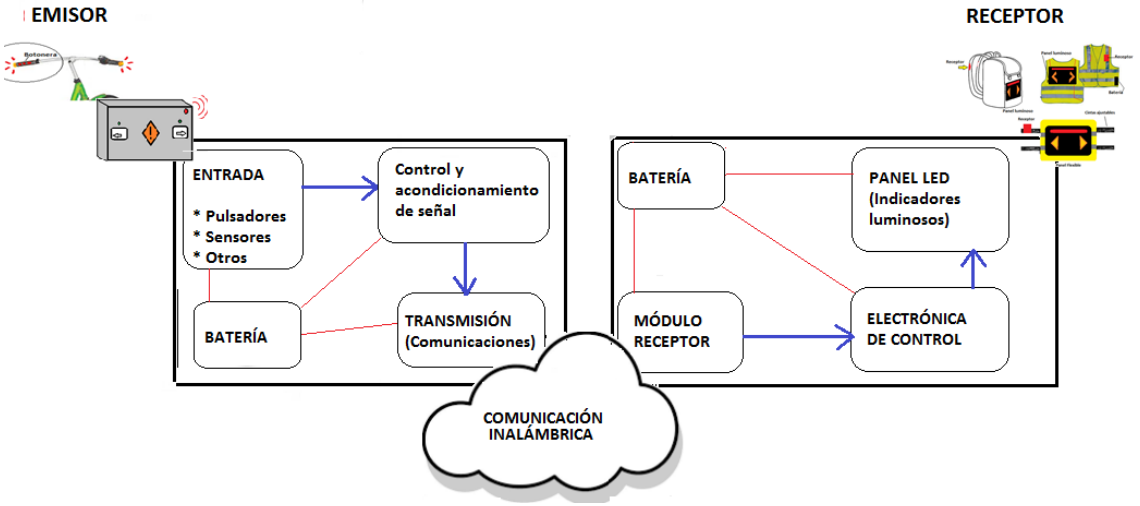


Ilustración 9. Diagrama de bloques básico

3 ANÁLISIS DE LA SITUACIÓN ACTUAL.

Cuando surgió la idea inicial para la realización de este proyecto, apenas existían soluciones orientadas cubrir esta necesidad.

Los productos que encontramos eran proyectos personales de usuarios que compartían en la web, como un proyecto de e-textiles consistente en tejer leds en una chaqueta y utilizar lilypad – Arduino para su control[6]

Sin embargo, en los últimos años el crecimiento de los usuarios de bicicletas ha llevado asociado un aumento de las alternativas de producto disponibles.

3.1.1 USUARIOS

El aumento de nº de usuarios en núcleos urbanos, la apuesta de las instituciones por el fomento del uso de la bicicleta, y la falta de seguridad percibida por el usuario no sólo para su utilización sino para el cumplimiento de la normativa, indican que existe una necesidad no cubierta en el mercado.

Según la información del último *barómetro de la bicicleta* [7] publicado en España:

20,5 millones de españoles tienen bici para uso personal

cerca de 3 millones la utilizan casi a diario

Saben ir en bici más de 34 millones de españoles**	**Tienen bicicleta para uso personal** 20,5 millones de españoles	**Son usuarios con alguna frecuencia de la bicicleta** más de 15,7 millones de españoles	**Van en bicicleta cada día o casi cada día** cerca de 3 millones de españoles	**La última semana han ido en bicicleta** más de 9,5 millones de españoles

Ilustración 10. Barómetro de la bicicleta - DGT. Usuarios de bicicletas 2011

Adicionalmente se indica que:

- Ha aumentado el número de usuarios de bicicletas, principalmente entre la gente joven y aquellos que la utilizan para desplazamientos cotidianos en núcleos urbanos.
- Sólo un tercio de los usuarios reconocen circular por la calzada:
    - El mayor problema es la falta de seguridad percibida.

[6] http://www.instructables.com/id/turn-signal-biking-jacket/
[7] http://www.dgt.es/Galerias/seguridad-vial/investigacion/estudios-e-informes/INFORME-BAROMETRO-BICICLETA

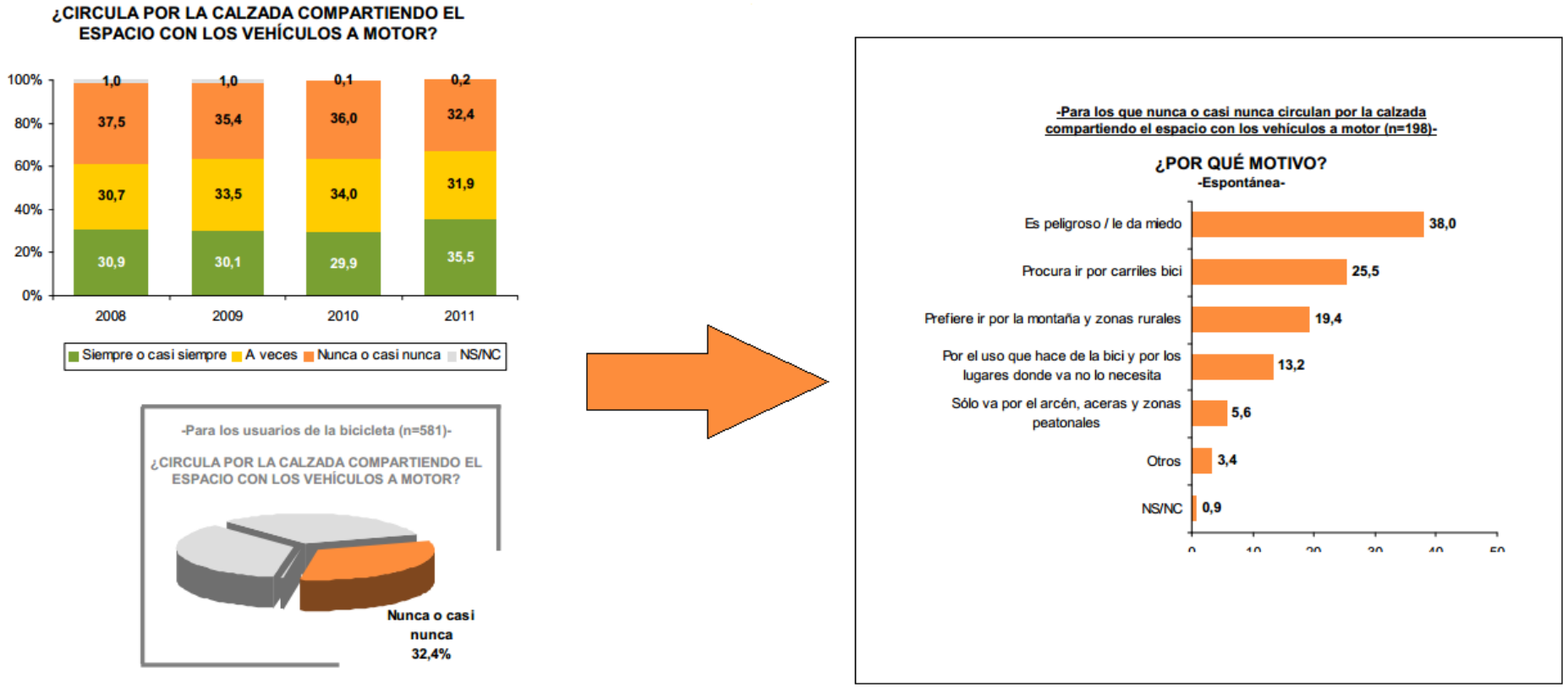


Ilustración 11. Estadísticas de uso de bicicleta - Circulación[8]

[8] http://www.dgt.es

3.1.2 PRODUCTOS SUSTITUTIVOS

Nos centraremos en aquellos dispositivos que incluyan indicadores de dirección / intermitentes, dado que existen multitud de dispositivos reflectantes para cubrir las necesidades de luz delantera y trasera y reflectantes para circulación nocturna.

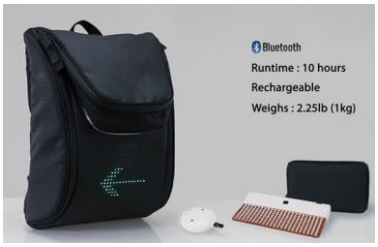
Tampoco incluiremos aquellos dispositivos cuya transmisión es por cable, es decir, no disponen de comunicación inalámbrica.

Emisor / Receptor - Inalámbrico

	Seil Bag	Bikeman	Bike Signals	Hazte visible	Bicygnals
Se comercializa	NO	SI	SI	SI	SI
Precio final	NO	54,00€ - 100,00 €	30,00 €	33,00 €	65,00 €
Receptor	Mochila	Varios soportes	Cinturón	Cinturón	Bajo sillín
Emisor	Manillar	Manillar	Manillar	Manillar	Manillar
inalámbrica	Bluetooth	2,4 GHz	2,4 GHz	2,4 GHz	2,4 GHz
Normativa color	No	No	Si	No	Si
Normativa equidistancia	No	No	Si	Si	Si
Escalable por usuario	Si	No	No	No	No
Alimentación emisor	Pilas	Pilas	Pilas	Pilas	Pilas
Alimentación receptor	Pilas	Batería recargable	Pilas	Pilas	Pilas

Tabla 2. Cuadro resumen productos sustitutivos

Seil Bag[9]



Proyecto 2010 – 2013 que finaliza por falta de interés de usuarios debido al alto coste del producto final.

Receptor: Panel led adaptable a Mochila / riñonera

Emisor: En manillar (botones derecho e Izquierdo)

Comunicación inalámbrico Bluetooth

Imagen: Monocromo

Alimentación: Pilas

Otros: Disponen de aplicación móvil para precargar imágenes diseñadas por el usuario

Desventajas: No cumple normativa respecto a color de intermitencias, y equidistancia de las luces respecto al eje. Unido a la precarga de cualquier tipo de imagen desde el móvil, puede confundir al resto de usuarios de la vía y causar el efecto contrario al perseguido: Falta de seguridad en las vías.

[9] http://www.leemyungsu.com/works.html

Bikeman[10]

Producto final comercializable desde 2014

A la venta en diversas plataformas web – China.

Receptor: adaptado a producto.

Emisor: En manillar

Comunicación inalámbrico 2,4 GHz / alcance máx. 3m.

Alimentación: Pila botón emisor, batería litio recargable receptor.

Precios[11]: Bajo sillín 54,00 €, panel trasero 73,98 €, mochila 99,98 €

Desventajas: Producto final cerrado a modificaciones de usuario. No cumple normativa respecto a color de intermitencias, equidistancia de luces respecto al eje…

Tecnología inalámbrica sensible a interferencias podría dar falsas indicaciones.

Bike Signals:

Receptor: cinturón con señales luminosas

Emisor: En manillar

Comunicación inalámbrica alcance máx. 3m

Imagen: Monocromo

Alimentación: Pilas

Precios[12]: 30,00 €

Desventajas: Producto final cerrado a modificaciones de usuario. Tecnología inalámbrica sensible a interferencias podría dar falsas indicaciones.

Hazte visible

Receptor: cinturón con 24 luces LEDs

Emisor: En manillar

Comunicación inalámbrica alcance máx. 3m

Imagen: Monocromo

Alimentación: Pilas

Precios[13]: 33,00 €

Desventajas: Producto final cerrado a modificaciones de usuario. Tecnología inalámbrica sensible a interferencias podría dar falsas indicaciones.

10http://www.bikeman.com.cn/
[11] www.lightinthebox.com
[12] http://www.amazon.co.uk/Bike-Signal-wireless-right-indicator/dp/B004Q5QD2E
[13] http://www.haztevisible.com/wp-content/uploads/2013/11/Manual-de-instrucciones-25-de-Nov.pdf

Bicygnals[14]

Receptor: Módulo luminoso bajo sillín

Emisor: En manillar – Incluye módulos de intermitencia

Comunicación inalámbrico 2,4 GHz

Imagen: Monocromo

Alimentación: Pilas tanto en emisor como en receptor

Precios[15]: 65,00 €

Desventajas: Producto final cerrado a modificaciones de usuario.

A pesar de que es muy fácil de transportar, no dispone de baterías o sistema de recarga adicional. Ambos módulos (emisor y receptor) funcionan mediante pilas.

Tecnología inalámbrica sensible a interferencias podría dar falsas indicaciones.

Accesorios adicionales.

Los accesorios como manillares con intermitentes integrados, guantes, cascos, u otras prendas son de interés dado que al ser nuestro producto escalable, y poder disponer de diversos receptores, es posible integrarlos en proyectos.

	Ventajas	**Productos similares**
Manillar	Incluir un módulo receptor en el manillar (equidistantes y elementos más externos del vehículo) aumenta la visibilidad y por tanto la seguridad del usuario.	Blinkergrips[16] 
Guantes	La señalización de giro mediante señales con los brazos es la norma internacional adoptada actualmente. Sin embargo, en condiciones de baja visibilidad no se percibe correctamente por el resto de usuarios de la vía. Disponer de elementos que hagan aumenten la visibilidad implica un aumento de la seguridad del usuario.	Zackees 
Retrovisores	Permite al usuario no sólo ser más visible, sino tener un mayor control sobre el resto de vehículos mediante el aporte de retrovisores.	Winkuu[17] 

Tabla 3. Accesorios: retrovisor, manillar y guante.

[14] http://www.bicygnals.co.nz/press.html
[15] http://www.haztevisible.com/wp-content/uploads/2013/11/Manual-de-instrucciones-25-de-Nov.pdf
[16] http://blinkergrips.com/
[17] http://www.winkku.co.uk/three.php

4 DISEÑO DEL SISTEMA

Se plantea un producto que consta de un **sistema 1** en el manillar de la bicicleta, que permite la entrada de instrucciones mediante pulsadores o elementos similares, y que tras el procesado de las señales las envía a un **sistema 2** móvil y adaptable que lleva el propio usuario. El sistema 2 recibe las instrucciones de forma inalámbrica y procede al encendido / apagado de los leds.

En el diagrama de bloques que se adjunta a continuación se muestra el esquema propuesto con sus respectivos módulos:

NODO

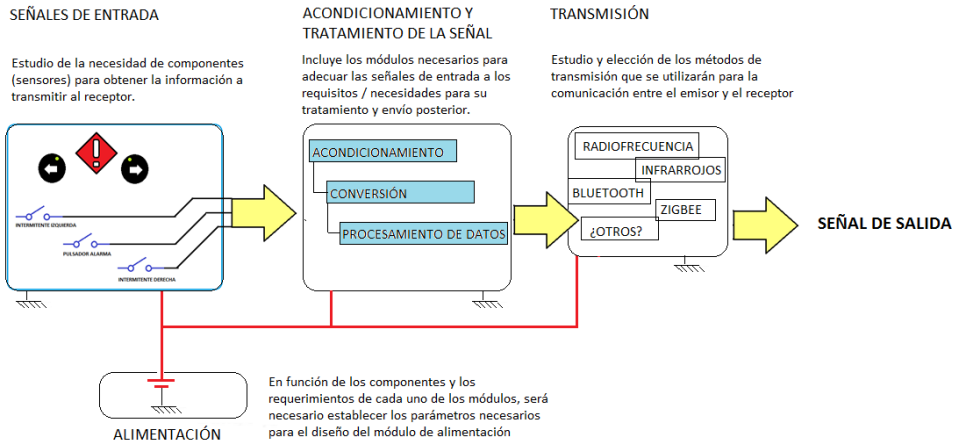


Ilustración 12. Introducción al NODO

COORDINADOR

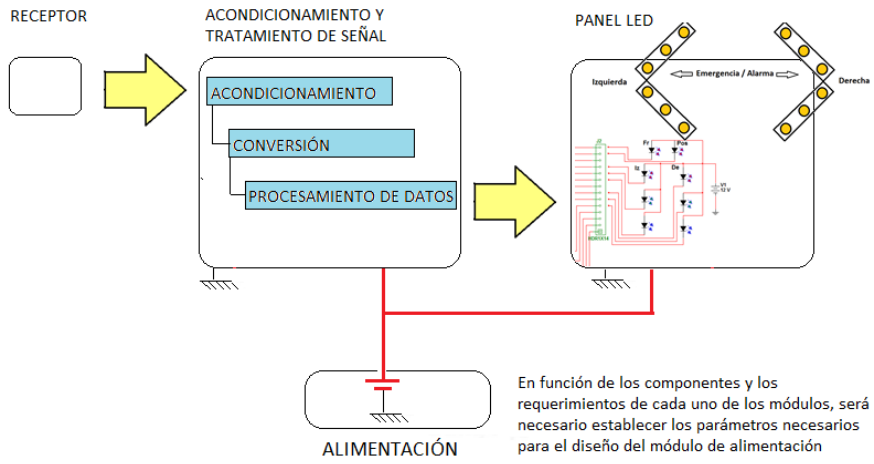


Ilustración 13. Introducción al COORDINADOR

4.1 ELEMENTOS DE ENTRADA: SENSORES

Los sensores son elementos que utilizamos para medir variables físicas de interés.

*Se denomina **transductor** a todo dispositivo que convierte una señal de una forma física en una señal correspondiente pero de otra forma física distinta. Es por tanto un dispositivo que convierte un tipo de energía en otro.*

*Un **sensor** es un dispositivo que, a partir de la energía del medio donde se mide, da una señal de salida transducible que es función de la variable medida.*

Sensores acondicionadores de señal
Ramón Pallás Areny

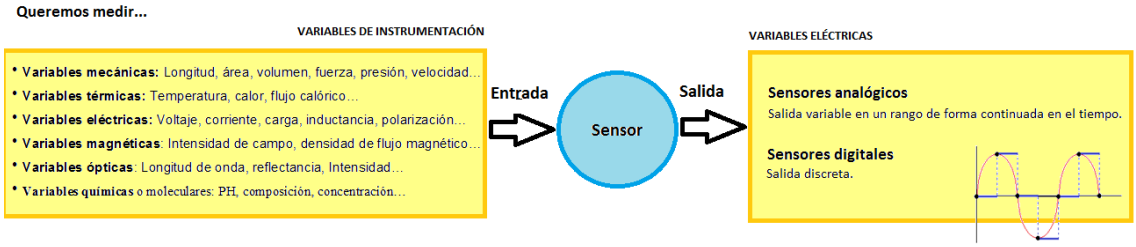


Ilustración 14. Sensores

Existen diferentes tipos de sensores y clasificaciones en función de las características a las que atendamos. Generalmente se dividen en dos grandes grupos atendiendo al tipo de salida:

- **Sensores analógicos**: salida variable dentro de un rango. Generalmente les afecta el ruido, interferencias, distorsiones… Por lo que para trabajar con ellos se hace necesario el diseño de circuitos acondicionadores de señal. En nuestro caso usamos para indicar derecha/izquierda un joystick con salida analógica.
- **Sensores digitales**: Salida discreta. Proporcionan una salida binaria (0 o 1) y en nuestro sistema los utilizamos para las entradas de la alarma y el freno.

  Dentro de los sensores digitales disponibles, por su sencillez, su bajo coste, y dado que cumple con los requisitos para el diseño, se plantea la utilización de:

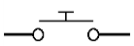
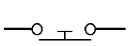
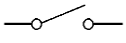
	Abierto	Cerrado	Funcionamiento básico
Pulsador			Al accionarlo permite el paso de corriente de forma **temporal** ya que al soltarse, se interrumpe volviendo al estado inicial.
Interruptor			Al accionarlo abre o cierra un circuito de forma **permanente** hasta que se vuelve a accionar.

Tabla 4. Pulsador vs Interruptor

El comportamiento objetivo es que el usuario realice una pulsación para activar la señalización que corresponda, y posteriormente vuelva a pulsar una vez realizada la maniobra para desactivarla.

Existen diversos actuadores que permiten este comportamiento. En nuestro caso hemos escogido dos con diferentes comportamientos:

Indicación	**Sensor escogido**	
Giro a la derecha	Joystick	
Giro a la izquierda	Joystick	
Señal de alarma	Pulsador	
Señal de freno	Pulsador	

Tabla 5. Actuadores escogidos.

4.2 ACONDICIONAMIENTO Y TRATAMIENTO DE LA SEÑAL

En función de los datos de salida de los sensores se debe acondicionar la señal para poder interpretarla y procesarla en las siguientes etapas.

¿Cuáles son los motivos?

Podríamos necesitar linealizar una señal, convertirla de analógica a digital, aplicar filtros, precisar amplificación debido a que es demasiado pequeña o necesitar un cambio de magnitud para su tratamiento, entre otras cosas.

Esto es lo que conocemos como **acondicionamiento de la señal**.

En nuestro caso, disponemos de:

- Interruptor y pulsador para señalizar alarma y freno, cuya señal es del tipo:

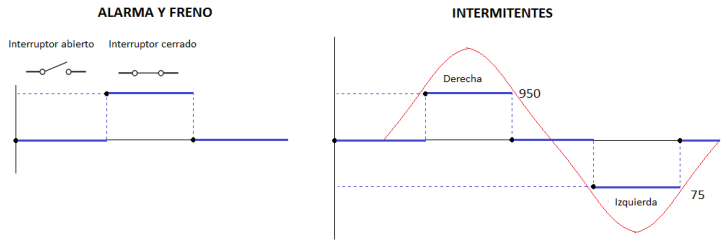


Ilustración 15. Interpretación de actuadores.

Hemos revisado posibles rebotes de la señal que pudieran afectar al funcionamiento del circuito, solventándose cualquier anomalía en la señal mediante la programación del controlador.

Además, como se verá a continuación, se ha optado por una solución basada en el uso de microcontroladores. La señal se conecta a los mismos mediante “puertos” que generalmente cuentan con acondicionamiento de señal y protección para prevenir daños en el sistema del microprocesador.

Por lo tanto en nuestro caso particular no es preciso acondicionar la señal; **nos centraremos en la etapa de procesado y control**.

Si en desarrollos futuros se optara por incluir módulos de entrada con acelerómetros (generalmente requieren amplificador de señal), o sensores lumínicos, sería necesario incluir una etapa de acondicionamiento.

4.2.1 CONTROLADORES

En la elección de la electrónica para el control del funcionamiento de la señalización podemos optar por dos soluciones muy distintas.

1. Puertas lógicas.
2. Microcontroladores.

4.2.1.1 PUERTAS LÓGICAS

La primera solución está basada en el desarrollo de una electrónica específica con puestas lógicas, donde las instrucciones del comportamiento de la señalización estén directamente implementadas en hardware.

Esta solución presenta algunas ventajas, como el coste bajo y mayor facilidad para disminuir consumos.

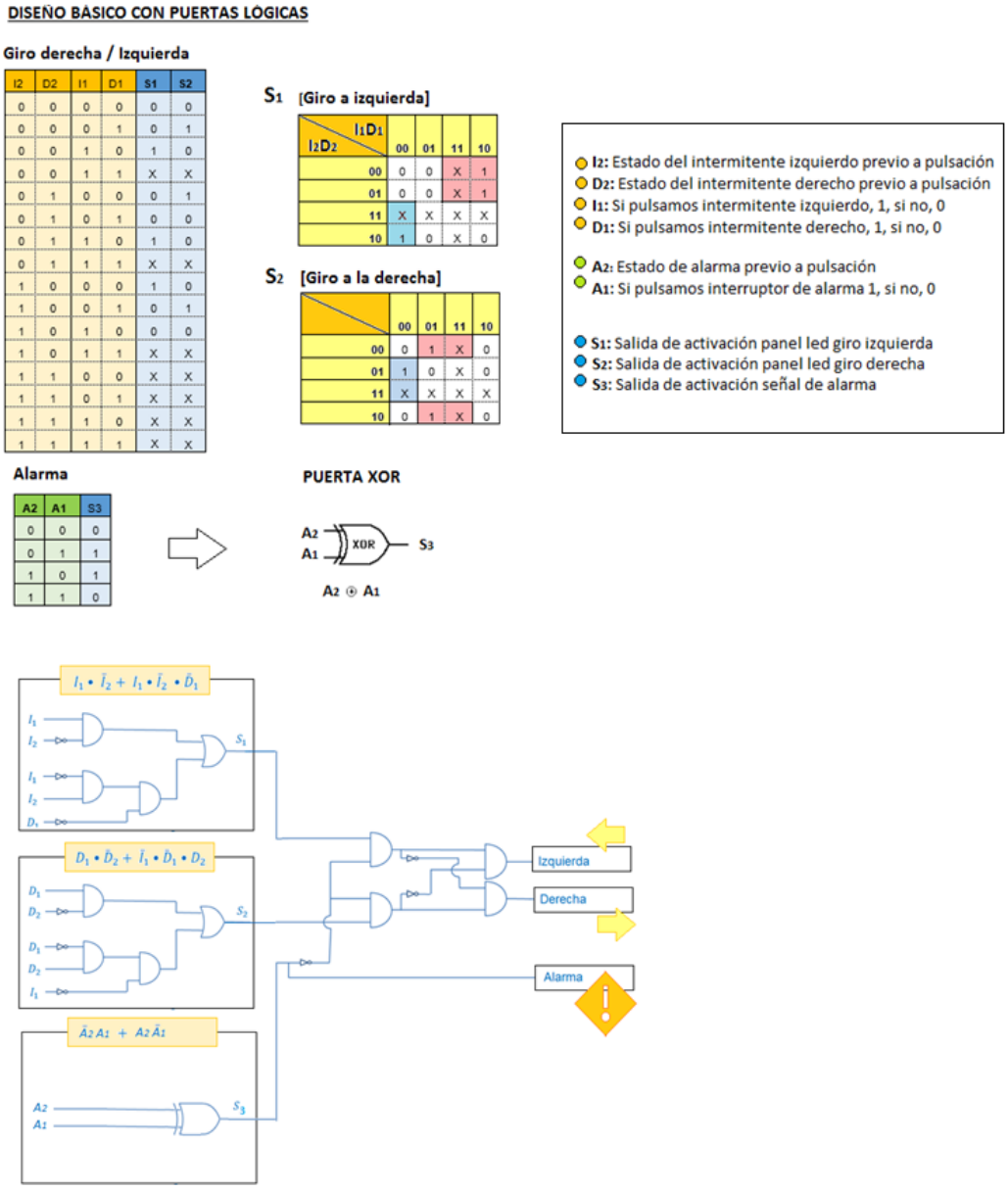


Ilustración 16. Diseño con puertas lógicas

4.2.1.2 MICROCONTROLADORES

Los microcontroladores generaron una revolución en la forma de pensar y diseñar circuitos electrónicos. Además, los microcontroladores poseen muchas ventajas respecto a la lógica cableada y a la lógica programada, debido a que tienen bajo costo, alta inmunidad al ruido eléctrico y pequeño tamaño…

Microcontroladores. Funcionamiento, programación y aplicaciones prácticas.
www.FreeLibros.me

La segunda solución se basa en usar una electrónica de propósito general e implementar en software las instrucciones sobre el comportamiento de la señalización.

El uso de microcontroladores tiene la ventaja de la flexibilidad a la hora de actualizar y mejorar el producto. A pesar de que el coste del hardware en nuestro caso particular es superior al desarrollo basado en puertas lógicas, los precios de estos componentes han bajado mucho y su evolución prevista es continuar con dicha tendencia.

En el diagrama de bloques de la figura se muestra la situación de los sistemas de control de la señalización propuestos para el desarrollo del proyecto.

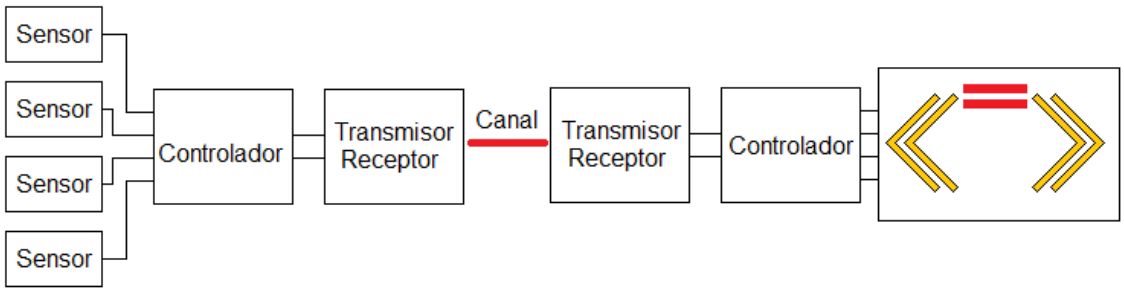


Ilustración 17. Diagrama de bloques – Sistemas de control.

El uso de estos componentes nos ofrece la ventaja de que podemos optar por que los controladores en el transmisor y el receptor sean iguales en hardware y sólo diferentes en software.

El estado de la técnica actual permite al usuario tanto montar su propio sistema a partir de los componentes (**microcontrolador con hardware especifico**) o adquirir uno circuito integrado que se ajuste a sus necesidades (**microcontrolador de propósito general**). Lo estudiaremos a continuación

Microprocesadores vs Microcontroladores

Un *microprocesador* es un circuito integrado que realiza diferentes tareas gracias a que contiene en su interior circuitos complejos creados a partir de transistores, actuando como unidad de procesamiento central (CPU) de los microcontroladores.

No están preparados para comunicarse con los dispositivos periféricos que se les conectan, por lo que necesitan de otros componentes adicionales para su funcionamiento.

Por tanto, los microprocesadores forman parte de un sistema mayor: el microcontrolador.

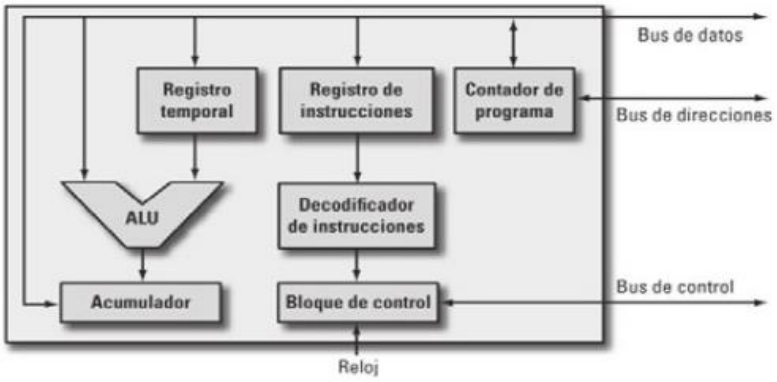


Ilustración 18. Componentes de un microprocesador[18].

Un *microcontrolador* es un sistema completo, autónomo e independiente que se diseña para que todos los componentes que lo conforman estén integrados en el mismo chip.

Para el diseño de un microcontrolador es necesario tener una imagen de los distintos componentes que lo conforman.

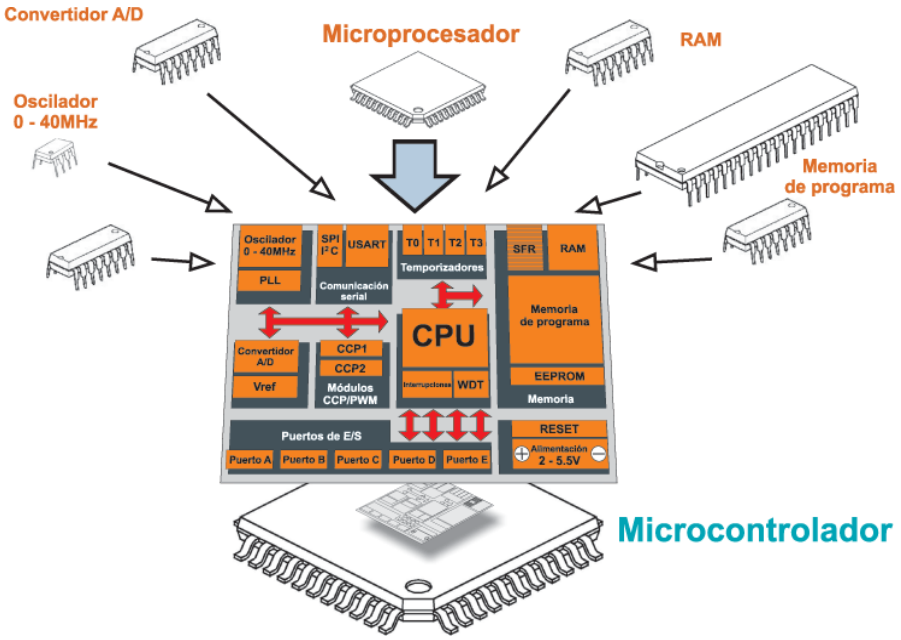


Ilustración 19. Microcontroladores[19]

Uno de los primeros controladores empleados y de amplia difusión han sido los PIC o “Controladores de Interfaz Periférico”.

[18] Fuente: Microcontroladores: Funcionamiento, programación y aplicaciones básicas. http://es.slideshare.net/aastuhuamanruiz/microcontroladores-35940999
[19] http://tic-tac.teleco.uvigo.es/profiles/blogs/microcontroladores-vs-microprocesadores

"Arduino" - Microcontrolador de propósito general

"El problema que hay es que por culpa del sistema de patentes se cerraba la posibilidad para mucha gente de aprender cómo funcionaban las cosas…

… El hardware abierto significa volver a tener la posibilidad de mirar que hay dentro de las cosas, pero hacerlo desde una forma que esté permitido, que sea éticamente correcto, que sea legal y que permita mejorar la educación."

David Cuartielles – Arduino The Documentary

Tratando el concepto del desarrollo hardware desde un nivel superior, se evalúa partir de una placa que ya contenga todos los componentes integrados incluyendo el microcontrolador.

Para su elección, nos decantamos por aquellas que estén basadas en plataformas de hardware libre, que sean funcionales y económicas, y cuyo entorno de desarrollo sea sencillo y de amplia difusión.

Adicionalmente, se considera una ventaja que cuente con elementos disponibles para la futura evolución del producto y su integración en proyectos multidisciplinares, según los requerimientos del mercado.

Al evaluar las alternativas presentes en el mercado, encontramos un dispositivo llamado Arduino que es ampliamente utilizado en diversos tipos de aplicaciones, por ejemplo las domóticas.

Arduino nace en 2005, como un proyecto dirigido principalmente a estudiantes. En poco tiempo se abre su campo de aplicación al público en general.

Originalmente basado en un PIC de 8 bits, desde octubre de 2012 se usa también con microcontroladores CortexM3 de ARM de 32 bits.

¿Por qué Arduino?

- Producto asequible, multiplataforma y que cuenta con amplia documentación.
- Entorno de programación sencillo (Processing / Wiring). No requiere de tarjetas de programación (la placa se conecta vía serie al ordenador por cable USB)
- Proyecto Opensource: Software y Hardware.
- Librerías disponibles para acoplar componentes externos (Ej. EEPROM; Ethernet, GSM, Wi-Fi …)
- Dispone de diferentes placas en función de las necesidades de desarrollo. (Arduino Uno R3, Arduino DUE, Arduino Yún, Arduino Mega ADK, Arduino NANO y Mini, Arduino Leonardo…)

Ilustración 20. Arduino UNO y Arduino Leonardo

- Periféricos: Placas adicionales fácilmente acoplables, tanto de Arduino cómo de otras marcas (Xbee, Teclados, LCD, sensores digitales, SD-Card…)

- Comunidad de desarrolladores: Permiten el libre intercambio de códigos, diseños, tutoriales y proyectos, así como la constante evolución y mejora.

4.2.2 CONCLUSIONES

A pesar de que el diseño mediante puertas lógicas ofrece una ventaja debido a su sencillez, y ahorro en costes y consumo, nos encontramos con una gran restricción respecto a la flexibilidad: Cualquier nueva implementación o modificación requiere de desarrollos específicos y cambios completos en el diseño.

El diseño de un microcontrolador mediante la elección de sus componentes tiene como ventaja la realización de una placa completamente a medida, que pudiera suponer un ligero ahorro de costes, a costa de sacrificar ventajas que ofrece el sistema Arduino como el disponer de una comunidad de desarrolladores a nivel mundial que permiten la evolución constante no sólo del producto sino de los proyectos realizados a partir de Arduino.

Dado que este proyecto se entiende como un prototipo con una aplicación directa por parte de los usuarios, y con grandes posibilidades de mejora y desarrollos futuros, y por las ventajas que ofrece en cuanto a tecnología e integración, **nuestro sistema de control estará conformado por placas Arduino**, tanto en el bloque emisor como en el receptor.

4.3 MÓDULO DE COMUNICACIÓN: TRANSMISIÓN Y RECEPCIÓN

4.3.1 CONCEPTOS BÁSICOS

La comunicación se define como la transmisión de información de un lugar a otro.

Aplicado a la electrónica y a las telecomunicaciones, es el proceso por el que se envía, propaga y recibe una señal.

Dicha señal puede ser analógica o digital y se denomina canal al medio empleado para su transporte.

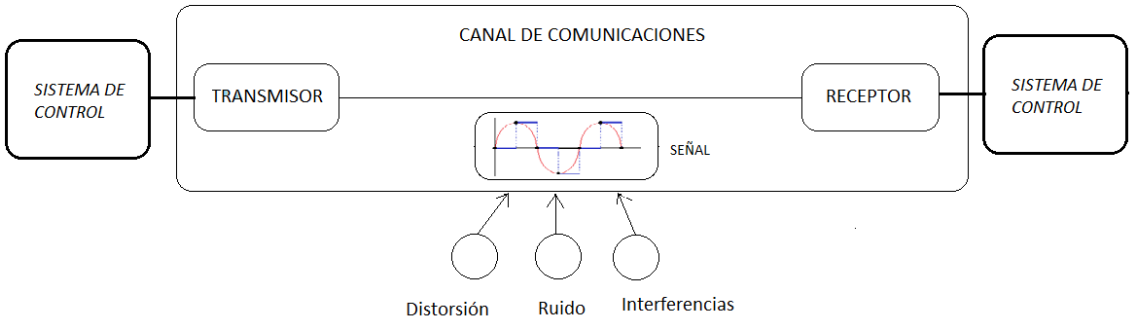


Ilustración 21. Comunicaciones

Definiciones:

Transmisor:	Transforma y adapta la señal recibida desde la fuente (sensores) a las condiciones del canal.
Receptor:	Transforma las señales recibidas para proporcionárselas al destino, en nuestro caso el módulo de iluminación.
Señal (Digital):	Información intercambiada
Elementos externos:	Ruido, Distorsión e interferencias
Canal de comunicaciones:	Entre los sistemas de control (emisor y receptor), existe el denominado canal de comunicaciones o medio por el que se transmite la información. *La elección del canal de comunicaciones resultará determinante* a la hora de fijar las condiciones de funcionamiento del sistema y sus grados de libertad.

Tabla 6. Conceptos Básicos Canal de Comunicaciones

4.3.2 CANAL DE COMUNICACIONES: TIPOS

Canal es el medio por el que se propaga la información.

Debe estar adaptado a la forma del transmisor y a la forma del receptor para optimizar el uso de energía:

"El mejor canal es aquel que requiere menos esfuerzo, es decir, aquel en el intercambio eficiente de la información implica un consumo mínimo de energía"

En función del criterio predominante para la elección de materiales, se elegirá el sistema de comunicaciones. De tal forma que si el factor determinante es el precio, el sistema más económico será el que tenga un canal de comunicaciones más simple y más barato.

Por contra, si dejamos al margen el factor económico, entran en juego otros canales de comunicaciones que permiten fundamentalmente mayor movilidad y ampliación de las funcionalidades.

Por comodidad, funcionalidades y escalabilidad, aquellos sistemas de comunicación inalámbricos (pese a llevar un mayor coste asociado) son los preferidos por los usuarios finales frente a aquellos que requieren de cableado adicional, aportando un elemento diferenciador frente a otros productos similares existentes en el mercado

A continuación analizaremos los siguientes tipos de canales, según el medio por el que se transmiten:

- Medio físico: Mediante una conexión con cable.
- Medio inalámbrico: Analizaremos los canales
    - Bluetooth y Zigbee (Radiofrecuencia)
    - Infrarrojos y Fototransistores (Luminosos)

4.3.2.1 MEDIO FÍSICO

Podríamos conectar los sensores mediante un cable físico de cobre directamente al sistema de señalización.

Si bien conllevaría un ahorro en costes en componentes y tecnología necesaria, simplificando el producto, cuenta con grandes desventajas:

- Alta tasa de fallos debido a la rotura del cable, o por fallos en el conector o por accidentes.
- Instalación más compleja y pérdida de movilidad.
- Limitación en integraciones futuras.

Dentro de las posibilidades que se presentan, habría dos que se podrían considerar.

- La primera es usar un hilo para cada señal a transmitir desde los sensores hasta el sistema de control de la señalización, más uno común.
- La segunda (*Ilustración 20*) es usar un solo par y realizar una asignación o partición del ancho de banda. Es lo que se denomina señalización por multifrecuencia. La presencia de una determinada frecuencia significa la activación de un sensor.

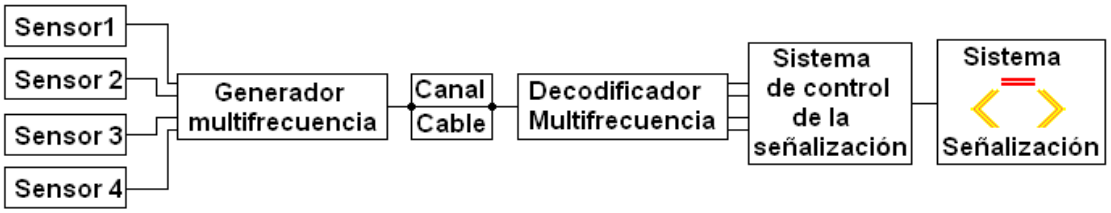


Ilustración 22. Esquema cableado con asignación de ancho de banda

4.3.2.2 MEDIO INALÁMBRICO

En la figura se muestra el espectro usado para la transmisión inalámbrica.

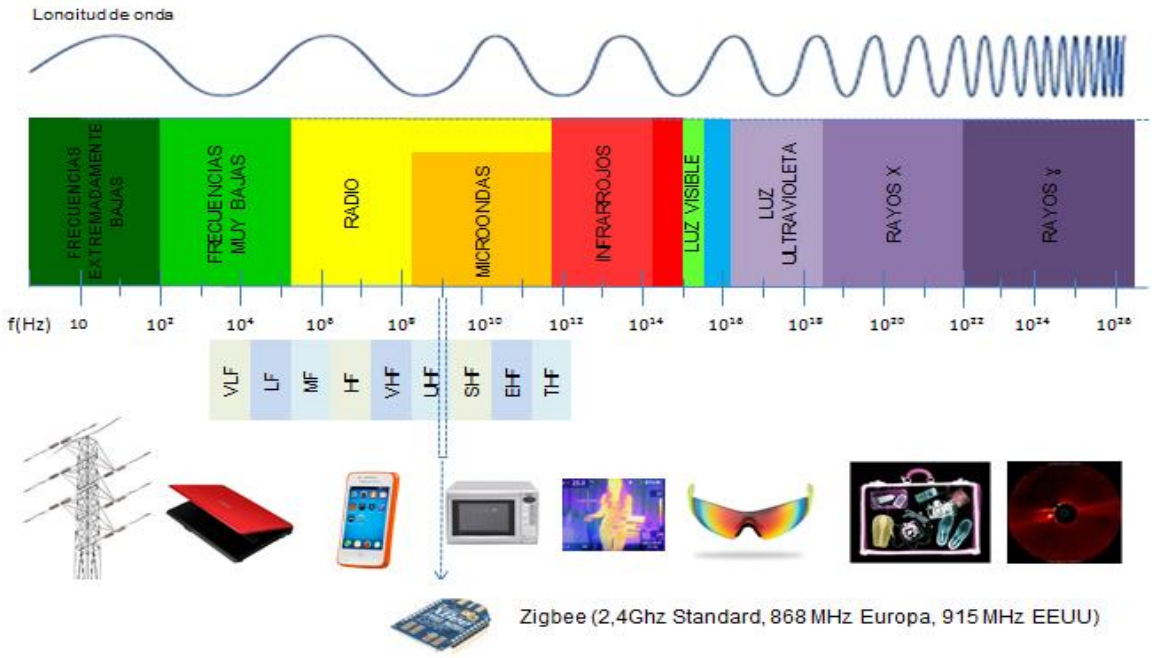


Ilustración 23. Espectro usado para la transmisión inalámbrica (Hz).

El encargado de adaptar la información al medio se denomina *Transceptor*, compuesto por un transmisor y por un receptor. Entre los más usados nos centraremos en los propuestos a continuación:

4.3.2.2.1 Radiofrecuencia.

Radiofrecuencia o RF	
Canal	Ondas electromagnéticas
Espectro	no visible
Frecuencia	3 Hz y 300 GHz
Canales propuestos	RF Bluetooth Zigbee

La información va modulando en amplitud (On-Off) las ondas electromagnéticas, una portadora RF con datos digitales, utilizando la frecuencia de 433,92 Mhz, como se indica en la figura. La Normativa Europea ETS 300 220 y ETS 300 683 (Compatibilidad Electromagnética), establece las condiciones y especificaciones técnicas.

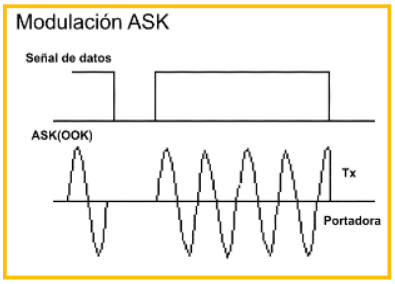


Ilustración 24. Modulación ASK

Es un sistema que ha proliferado fundamentalmente por su largo alcance, dado que con 2mWatt en antena tiene un alcance la transmisión de datos de varias decenas de metros.

Tiene la limitación de que la presencia de señales en el mismo canal, hace que la señal útil sea interferida y por lo tanto no recibida en destino.

También se utiliza la banda libre de 2,4 GHz, aunque el alcance es mucho menor.

Canal Bluetooth .....Radiofrecuencia

El canal es radiofrecuencia en la banda de 2,4 GHz. Sigue las especificaciones definidas en la industria para redes inalámbricas de carácter Personal (WPAN). Están pensadas para unir y sincronizar equipos dotados de microprocesadores y microcontroladores. Con 2 mW en antena se puede tener un alcance de 3 a 5 metros.

El hardware que compone el dispositivo Bluetooth está compuesto por dos partes:

- Un dispositivo de radio, encargado de modular y transmitir la señal.
- Un controlador digital, compuesto por una CPU, las interfaces con el dispositivo anfitrión, y un procesador de señales digitales (DSP - Digital Signal Processor) llamado Controlador de Enlace.

Canal ZigBee.....Radiofrecuencia

Está especializado en la transmisión de información de banda estrecha, como los datos de alarmas, sensores, etc...

La especificación ZigBee es un conjunto de protocolos para comunicaciones radio, basada en el estándar IEEE 802.15.4 (WPAN). Se emplean en aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

La normativa ZigBee utiliza la banda ISM; en concreto, 868 MHz en Europa, 915 en Estados Unidos y 2,4 GHz en todo el mundo.

La diferencia con el Bluetooth es que se pueden crear estructuras mucho mayores, con muchos más dispositivos (hasta 255 redes, cada una de ellas con 255 dispositivos) y consume mucho menos en reposo. Sin embargo su velocidad de transmisión de información es mucho menor con ZigBee que con Bluetooth.

4.3.2.2.1 Luminosos.

Un sistema de transmisión de datos por IR se basa en la emisión de ondas electromagnéticas en el espectro infrarrojo mediante un diodo IR y su recepción por un sensor IR, siendo el canal de transmisión el aire.

Luminosos	
Canal	Luz
Espectro	visible
Canales propuestos	Infrarrojos Foto transmisores

Tanto el transmisor como el receptor son elementos generadores y detectores de luz. La información se manda modulando su amplitud.

Infrarrojos..... Luminosos

La radiación infrarroja se encuentra comprendida entre el espectro visible y las microondas. Su longitud de onda es mayor que la de la luz visible.

Su fuente primaria es el calor o radiación térmica, de tal forma que cualquier objeto que tenga una temperatura superior al cero absoluto irradia ondas en la banda infrarroja.

Es el sistema comunicación óptica más difundido, y podemos encontrarlo en multitud de aplicaciones domésticas como por ejemplo en mandos a distancia (electrodomésticos, televisores, equipos de música, etc.) que modulan el transmisor IR con una frecuencia fija, que previamente contiene la información modulada en amplitud.

Precisa de la utilización de filtros (ópticos, paso banda…) para eliminar las posibles fuentes de ruido, ya que los receptores reciben interferencias ópticas o electromagnéticas debidas a que todas las fuentes de luz con emisiones en el espectro del ancho de banda del receptor pueden ser consideradas fuentes de ruido.

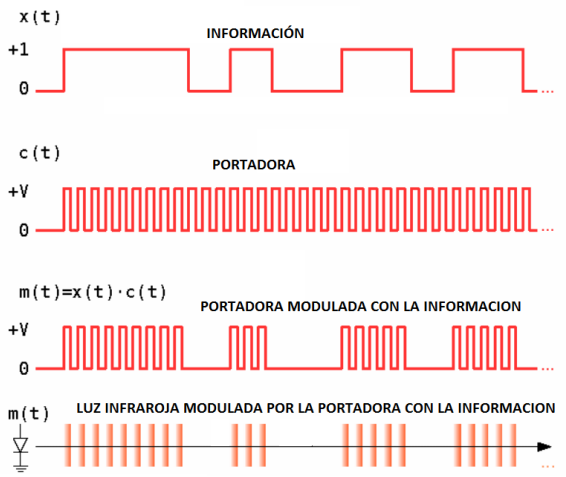


Ilustración 25. Modulación de señal

La información se codifica en multifrecuencia, y esta señal es la que modula a los transmisores de IR. El receptor debe decodificar la señal detectada.

Fototransmisores ..... Luminosos.

Fotoemisores: Transforman energía eléctrica en lumínica. Son dispositivos que al ser sometidos a una corriente o diferencia de potencial generan un haz de luz.
Para que un componente emisor de luz pueda ser utilizado en la transmisión de la información debe cumplir:

- El haz producido debe ser monocromático.
- La potencia óptica tiene que poder ser modulada por medios electrónicos.
- La respuesta debe ser rápida.

Los emisores más empleados en comunicaciones ópticas son los diodos emisores de luz (LED) y los diodos láser (LD). Son dispositivos optoelectrónicos semiconductores que operan en el infrarrojo próximo.

Fotodetectores: Transforman las ondas electromagnéticas recibidas a una señal eléctrica.
De entre los fotodetectores disponibles, destacar:

- Fotoresistencias (LDR): resistencia variable según la luz que incide. A mayor luz, menor resistencia nominal.
- Fototransistores: Son los fotodetectores más utilizados.
- Fotointerruptores: Se trata de una combinación de un led y un fototransistor alineados cuya función es la de funcionar como interruptor.

La incorporación de fototransmisores como módulo al proyecto no es inmediata, habría que desarrollar una electrónica adecuada ya que dichos elementos precisan de módulos de acondicionamiento de señal, y son sensibles al ruido. Por lo tanto descartamos su uso.

4.3.3 CONCLUSIONES

Tras analizar los diferentes canales y elementos de comunicaciones según se indica en la tabla 5, se procede a seleccionar el canal ZigBee:

Medio		**Motivos de elección / descarte**	**Resultado**
Físico	CABLE	Por motivos de comodidad, aceptación del usuario, movilidad y diseño.	DESCARTADO
Inalámbrico	LUZ • IR • Fototransmisores	Son dispositivos más sensibles al ruido y a los cambios en las condiciones externas (luz, temperatura, etc.), además de precisar desarrollos adicionales electrónicos para el acondicionamiento de la señal.	DESCARTADO
	RADIO • RF y Bluetooth	Se opta por el uso de tecnología Zigbee por: • Integración directa con Arduino • Está especializado en transmisión en banda estrecha • Basado en estándar • Comunicaciones seguras • Bajo consumo	DESCARTADO
	• Zigbee		**SELECCIONADO**

Tabla 7. Conclusiones – Selección canal de comunicaciones

4.4 MÓDULO DE SEÑALIZACIÓN

El módulo de señalización consiste en el conjunto de componentes y elementos necesarios para indicar al resto de usuarios las actuaciones que quiere realizar el ciclista, es decir, las instrucciones generadas originalmente a través del mando situado en el manillar.

Por consumo, tiempo de vida y versatilidad se ha optado por la utilización de diodos LED en lugar de cualquier otro tipo de emisor de luz.

Las siglas LED provienen del inglés Light Emitting Diode. (Diodo Emisor de Luz).Los Led son componentes capaces de emitir luz cuando circula corriente a través de ellos. Su funcionamiento es como el de los diodos (polarización en directa).

El color de la luz producida varía en función de la longitud de onda de emisión, que a su vez depende del material con el que está construido. Los colores más habituales son el rojo, verde o amarillo, aunque actualmente también se fabrican diodos azules e infrarrojos.

En nuestro caso existe una normativa referente al color de las luces (señalización) de los vehículos, por lo que es necesario tener en cuenta en el diseño que el color está asociado al tipo de información a transmitir, por ejemplo, los intermitentes son de color “ámbar”, las luces de freno son de color rojo, la luz de marcha atrás es de color Blanco, etc.

En nuestro caso, veremos dos tipos de diodos. Los diodos LED, los “Monocromo” y los diodos LED “RGB” que incorporan los tres colores (Rojo, Verde y Azul) en un solo LED.

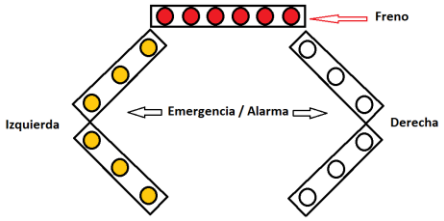


Ilustración 26. Módulo de señalización

- Para las indicaciones de **emergencia**, que coinciden con las luces de izquierda y derecha, el color que queremos es siempre el naranja. Por lo tanto emplearemos leds monocromo.
- La barra superior, que en nuestro proyecto es la luz que indica freno, es roja. Sin embargo, para facilitar implementaciones futuras, se ha empleado el uso de diodos RGB cuyo color dependerá de la programación realizada en el controlador. (Ej., si se desea encender una luz blanca.)

Desde el punto de vista del diseño es importante tener en cuenta:

1. La caída de tensión directa que hay que aplicar a cada diodo depende del color del diodo y oscila (ver tabla). Además, es necesario protegerlos de tensiones inversas que provocan roturas.

2. La intensidad luminosa del diodo depende de la corriente que circula por él, con dos efectos:
    a. Saturación, a partir de una determinada corriente la intensidad luminosa permanece prácticamente constante, con lo que aumenta el gasto pero disminuye la eficiencia.
    b. Los diodos LED no soportan corrientes elevadas, provocando su rotura.

En la siguiente tabla se indican las características de cada diodo en función de su color:

Color	V_f (V)	I_f (mA)
Rojo	1.85	30
Verde	2.20	25
Amarillo	1.80	50
Ámbar	1.80	50
Naranja	1.95	50
Azul	4.80	50

Tabla 8. Tensión y Corriente de un diodo en función del color.

4.5 MÓDULO DE ALIMENTACIÓN

4.5.1 BATERÍAS COMPACTAS VS PILAS ALCALINAS

Tanto las baterías como las pilas son dispositivos que almacenan energía. La diferencia fundamental es que mientras en las pilas se produce una reacción química electrolítica no reversible en las baterías o acumuladores dicha reacción es reversible, lo que implica que las primeras sean dispositivos NO recargables, mientras las segundas sí.

Es importante tener en cuenta que si bien las baterías pueden volver a cargarse conectándolas a una fuente de alimentación, el número de recargas no es infinito. Su vida útil se mide en ciclos de carga y una vez se alcanzan comienza a perder rendimiento de forma progresiva.

Esquema de funcionamiento:

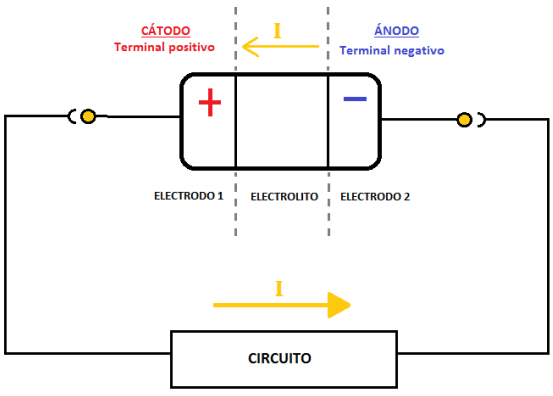


Ilustración 27. Batería - Esquema funcionamiento básico.

Electrodos:

- Ánodo (terminal negativo)
- Cátodo (terminal positivo)

Electrolito:

Medio físico para que la corriente circule por dentro de la batería. Puede ser un elemento líquido, sólido o una combinación de ambos.

	Ventajas	Desventajas
PILAS AA – AAA	Menor tamaño Menor coste de inversión inicial	No recargable Mayores residuos y contaminación Mayor coste a largo plazo
BATERÍAS	Producto ya integrado. Eco-friendly (Recargable) Menor coste a medio- largo plazo Mayor interoperabilidad con otros productos del mercado	Mayor coste de inversión inicial Mayor peso / tamaño

Tabla 9 Pilas vs Baterías

4.5.2 CONCLUSIONES

Hemos optado por la elección de baterías recargables para la integración en el producto final a las que integraremos una clavija de carga mini-usb o micro-usb estándar compatible con la mayoría de cargadores de dispositivos electrónicos existentes en el mercado que permitirá la interconexión tanto a la red eléctrica (enchufe) como recarga mediante la utilización de otros dispositivos móviles existentes en el mercado que mejorarán notablemente la autonomía del dispositivo como las celdas solares.

En la siguiente tabla se muestran los valores de tensión requeridos por los diferentes módulos empleados en nuestro desarrollo.

	Xbee	***Arduino***	***LEDs***	***Voltaje mínimo requerido***
Emisor	*2,8V – 3,3 V*	*$5V_{min}$ – $16V_{max}$*	*--*	*5V*
Receptor	*2,8V – 3,3 V*	*$5V_{min}$ – $16V_{max}$*	*$5V_{min}$ – $12V_{max}$*	*5V*

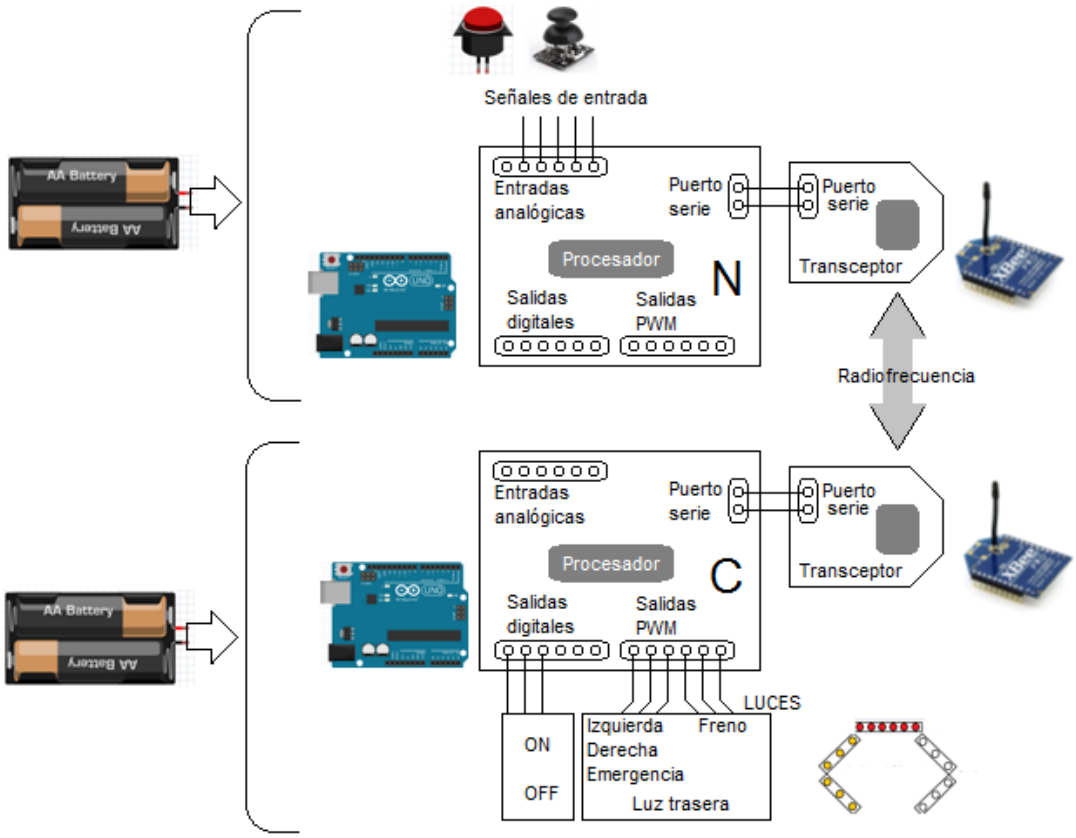
Tabla 10. Requisitos de voltaje Emisor - Receptor

5 MEMORIA DEL SISTEMA DE SEÑALIZACIÓN

En la Memoria del Sistema de Señalización se incluye la explicación de los trabajos realizados en configuración, integración y pruebas del sistema propuesto.

Para ello, una vez realizado el diseño cuyo esquema se muestra a continuación,

Esquema general



Bloque	**Elemento Utilizado**
Entradas	Pulsadores / interruptores
El controlador	Módulos Arduino tanto en el emisor como en el receptor
Transceptor de comunicaciones	Módulo Xbee conectado al puerto serie del controlador
Señalización	Diodos LED + circuito conmutador
Alimentación	Baterías recargables

Ilustración 28. Esquema general Sistema de Señalización

5.1 RESTRICCIONES BÁSICAS.

Denominaciones[20]:

- Al controlador + Xbee de la parte emisora le llamaremos: NODO
- Al controlador + Xbee de la parte receptora le llamaremos: COORDINADOR.

Las señales de entrada al nodo son las provenientes de los interruptores de los intermitentes y luces de emergencia y del pulsador del freno.

El NODO recoge el estado de las señales de entrada e informa al COORDINADOR y éste procede a encender las luces correspondientes a la señalización recibida.

Condición			
1	Software	Módulo de control	Diálogo esperado NODO - COORDINADOR
2	Hardware	Módulo transmisión	1 Nodo – 1 Coordinador
3	Software	Módulo de control	Estructura de trama.
4	Software	Módulo de control	Asignación de entradas.
5	Hardware	Módulo de control	Monitorización de control
6	Hardware	Módulo de control	Normativa de señalización

Tabla 11. Restricciones básicas.

Condición nº 1 - Diálogo esperado NODO - COORDINADOR

El diagrama de funcionamiento propuesto para el intercambio de información entre el NODO y el COORDINADOR es el que se muestra en la Ilustración 29.

Inicialmente se produce la sincronización entre ambos módulos, para proceder después al intercambio de órdenes.

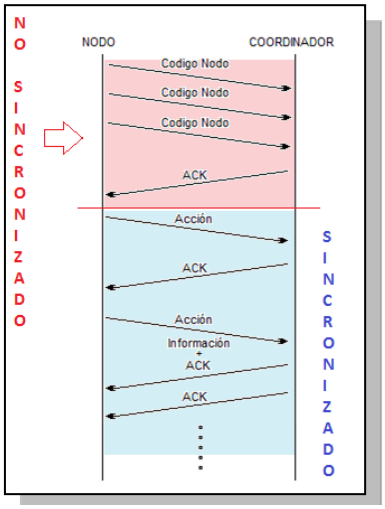


Ilustración 29. Comunicación NODO - COORDINADOR

Funcionamiento requerido:

- "**Código nodo**": Emisión de un código desde el nodo que recibe el coordinador. Dicho código es el que identifica el NODO y se definirá más adelante.
- "**ACK**": Es la secuencia que envía el COORDINADOR confirmando la recepción de la información que envía el NODO.
- "**Acción**": Cualquier información que requiere de una actuación por parte del módulo remoto.

[20] Hemos empleado la terminología utilizada en Xbee

Fases en las que pueden producirse las peticiones:

FASE I: Coordinador – Nodo no sincronizados

Se ha fijado que el NODO toma la iniciativa a la hora de sincronizarse enviando de forma cíclica su identificador y el COORDINADOR mantiene una escucha activa y cuando recibe la petición devuelve su código para confirmar la sincronización.

FASE II: Coordinador – Nodo sincronizados

Una vez producida la sincronización, entre el nodo y el coordinador hay un intercambio de palabras que contienen la información a tratar.

Condición nº 2 - 1 Nodo – 1 Coordinador

Se ha optado por la configuración punto a punto (1 NODO – 1 COORDINADOR) aunque en ZigBee el protocolo nos permite controlar en una misma red un gran número de dispositivos,

Como propuesta de mejora para desarrollos futuros se puede incluir el control de varios nodos con un único coordinador.

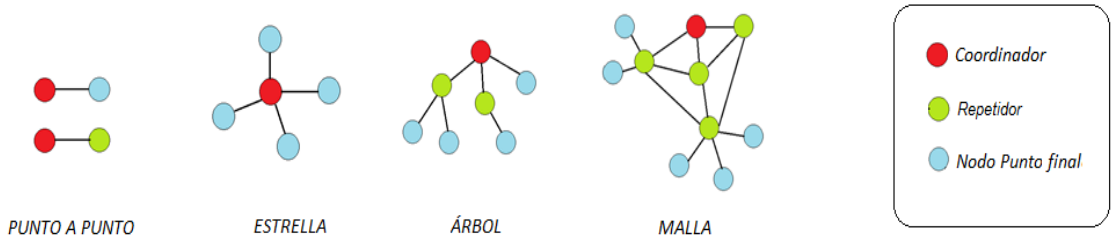


Ilustración 30. Tipos de redes Xbee.

Condición nº 3 - Estructura de trama.

Entre el NODO y el COORDINADOR el intercambio de información tiene una estructura previamente definida para que ambos puedan procesarla.

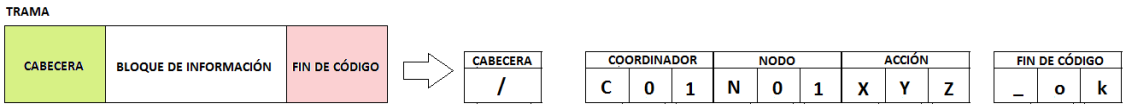


Ilustración 31. Estructura trama Nodo – Coordinador

Para el proyecto se ha elegido un código con la siguiente estructura:

	Descripción	**Nº**	**Carácter**
Cabecera	Por facilidad de programación he elegido sólo un carácter como bloque de inicio, de tal forma que dicho carácter cumpla que sea poco empleado.	1	"/"
Información	***Orden de trama:*** Los caracteres del identificativo del coordinador y los caracteres del identificativo del nodo, en este orden. ***Nº caracteres:*** tres caracteres alfanuméricos tanto para NODO como para COORDINADOR. Se ha configurado como código del nodo "N01" y un código de coordinador "C01".	9	Alfanumérico Restringido "/"
Fin de campo	Definido por un carácter, "guion bajo"	1	"_"
Fin de información	Indican el fin de la trama.	3	"_OK"

Tabla 12. Estructura de trama.

La respuesta que espera el NODO y por lo tanto lo que tiene que emitir el COORDINADOR es un código que denominaremos Indicativo Global. Se muestra la estructura propuesta:

Inicio + Código del enlace + orden [opcional] + "_ok"

Ejemplo: "/C01N01Fr1_ok"

La justificación de la estructura usada en este proyecto es por la presencia de "ruido", es decir, caracteres recibidos por un Arduino y no enviados por el otro.

Para evitar este ruido, se ha optado por que las rutinas de lectura desechen todos aquellos caracteres que lleguen antes del carácter de inicio.

Condición nº 4 - Asignación de entradas.

Las señales de entrada al nodo son independientes entre sí, es decir, cada señal ocupa una entrada diferente.

Se ha fijado la siguiente asignación a las entradas:

Entrada	Señaliza	Estado	Comportamiento
0	**Pulsador de freno** *El color es rojo.* *La luz es fija.*	**ON**	Enciende la luz de freno. Se puede dar simultáneamente con los intermitentes y con la luz de emergencia.
		OFF	Apaga la luz de freno.
1	**Intermitente izquierdo** *El color es ámbar.* *La luz es intermitente*.	**ON**	Enciente señal de giro a izquierda. Estado anterior: • Si estaba encendido el intermitente derecho lo apaga. • Si estaba encendida la señal de alarma, continúa la alarma.
		OFF	Apaga intermitente izquierdo
2	**Intermitente derecho** *El color es ámbar.* *La luz es intermitente*.	**ON**	Enciente señal de giro a la derecha. Estado anterior: • Si estaba encendido el intermitente izquierdo lo apaga. • Si estaba encendida la señal de alarma, continúa la alarma.
		OFF	Apaga intermitente derecho
3	**Luces de emergencia** *El color es ámbar.* *La luz es intermitente*.	**ON**	Enciende señal de alarma (luces de intermitente izquierdo y derecho simultáneamente). Estado anterior: • Si estaba encendido el intermitente izquierdo lo apaga. • Si estaba encendido el intermitente derecho lo apaga.

Tabla 13. Asignación de comportamiento de entradas.

Condición nº 5 – Monitorización de control.

El usuario cursa una orden en el manillar, y se efectúa un cambio en el panel luminoso trasero.

Para la señal de freno, el comportamiento es de encendido solo durante la pulsación, por lo que no existe necesidad de visualizar. Sin embargo, en el caso de las señales de intermitencia, el comportamiento es de doble pulsación: pulsa para encender y pulsa para apagar.

Por lo tanto, el usuario debe conocer en todo momento el estado de estas señales para evitar errores (ejemplo, que quiera encender derecha, ya esté encendida y la apague).

Por este motivo, se estima incluir una monitorización mediante leds en el módulo que está en el manillar, es decir, en el NODO:

1 Led de SINCRONIZACIÓN: Encendido si existe sincronización entre el NODO y el COORDINADOR.

2 Led de Intermitente Izquierdo; encendido si la señal está activa.

3 Led de Intermitente Derecho; encendido si la señal está activa.

4 Led de Batería; encendido cuando el nivel de la batería es bajo.

Condición nº 6 – Normativa de señalización

Según la normativa existente, las señales luminosas en los vehículos deben cumplir:

- Intermitentes y emergencia de color “Ámbar”.
- Luz de Freno “Roja”.
- Señales de intermitencia ubicadas de forma simétrica respecto al eje central.

5.2 HARDWARE

5.2.1 PULSADORES

GIROS..... Pulsadores

Para el giro derecha / izquierda se ha utilizado un módulo Joystick[21].

Ilustración 32. Elemento Joystick

En el caso de que se desease emplear un pulsador con un hardware diferente para obtener el funcionamiento descrito en la tabla, no requerirá reprogramaciones en nuestra placa de control Arduino.

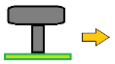
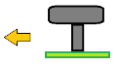
Estado inicial	Acción	Estado final
Reposo		Derecha ON
Derecha ON		Reposo
Izquierda ON		Derecha ON
Reposo		Izquierda ON
Derecha ON		Izquierda ON
Izquierda ON		Reposo

Tabla 14. Joystick – Pulsaciones vs Estado inicial y final de intermitencia

Indicar que se ha programado que tras una intermitencia de 20 segundos el dispositivo volverá al estado de reposo.

[21] PS2 Game Joystick Module For Arduino - http://www.banggood.com/5Pcs-PS2-Game-Joystick-Module-For-Arduino-p-951186.html

ALARMA ..... Pulsadores

Para la señal de alarma, se utilizará un pulsador independiente rojo.

Ilustración 33. Pulsador de Alarma

Indicar que el propio Joystick empleado para los giros ofrece la posibilidad de transmitir la señal de alarma, pero a pesar de que el uso de un pulsador adicional aumenta el coste del producto, también aumenta la seguridad del mismo.

Se ha considerado que un pulsador independiente, facilita el acceso y su uso en condiciones de "alarma" o situaciones críticas, por lo que finalmente nos decantamos por su utilización.

Estado inicial	Acción	Estado final
Reposo		Alarma ON
Alarma ON		Reposo
Izquierda ON		Izquierda OFF Alarma ON
Derecha ON		Derecha OFF Alarma ON

Tabla 15. Pulsador vs Estado inicial y final de alarma.

Indicar que esta señal no tendrá desactivación temporal, sólo se desactivará mediante una nueva pulsación.

FRENO ..... Pulsadores

La señal de freno se activará cuando el usuario pulse las manillas de freno del manillar del vehículo, y en nuestro proyecto se simulará mediante la presión de un pulsador.

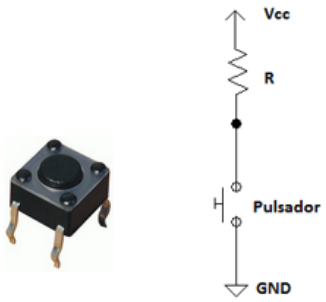


Ilustración 34. Configuración PULL UP

Durante el tiempo que permanezca presionado, se encenderán unas luces de freno compatibles y simultáneas con las indicaciones de giro y alarma.

En el momento en el que cesa la pulsación, la señal se apaga.

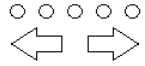
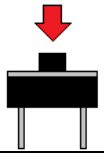
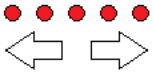
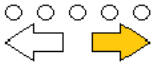
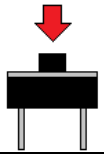
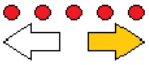
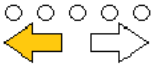
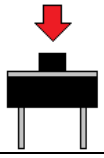
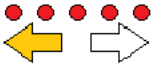
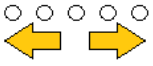
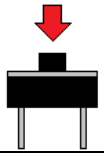
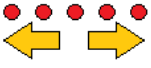
Estado inicial	Acción	Estado final
 Reposo		 Freno ON
 Derecha ON		 Freno + Derencha
 Izquierda ON		 Freno + Izquierda
 Alarma ON		 Freno + Alarma

Tabla 16. Pulsador vs Estado inicial y final de freno.

5.2.2 CONTROL: ARDUINO

El módulo de control es el encargado de procesar la información recibida de los sensores, para generar el protocolo de señalización adecuado.

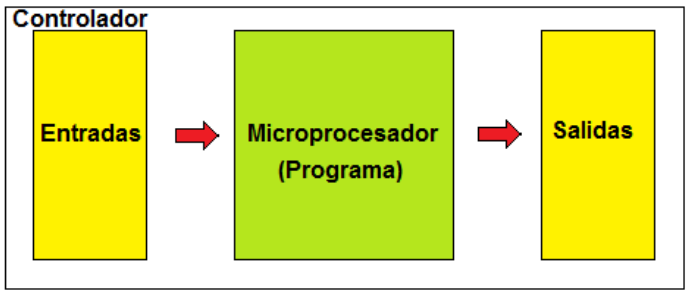


Ilustración 35. Diagrama bloques tipo Controlador

De todos los controladores existentes en el mercado, hemos elegido “Arduino” por ser de gran difusión (estandarización), por ser asequible, multiplataforma y por su fácil programación, además de ser Opensource y dar escalabilidad que permita integraciones futuras.

De las diferentes placas disponibles dentro de la gama de productos Arduino, empleamos el modelo “UNO”, aunque existen otras placas que mejoran tanto el tamaño como el coste.

En nuestro caso, nuestro módulo de control está compuesto por placas Arduino, que contienen la programación y procesan la información recibida por los transceptores.

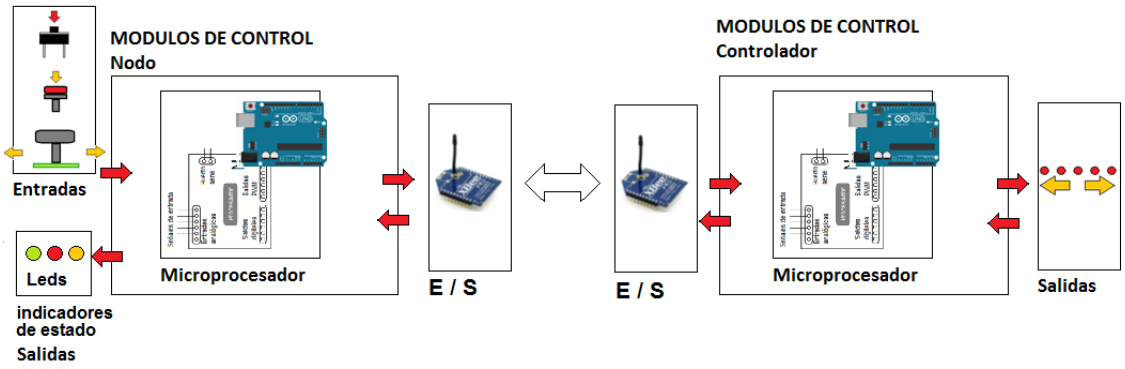


Ilustración 36. Esquema general - Bloques

Para configurar nuestro sistema, en el módulo Arduino disponemos de distintos terminales que gestionan las entradas / salidas.

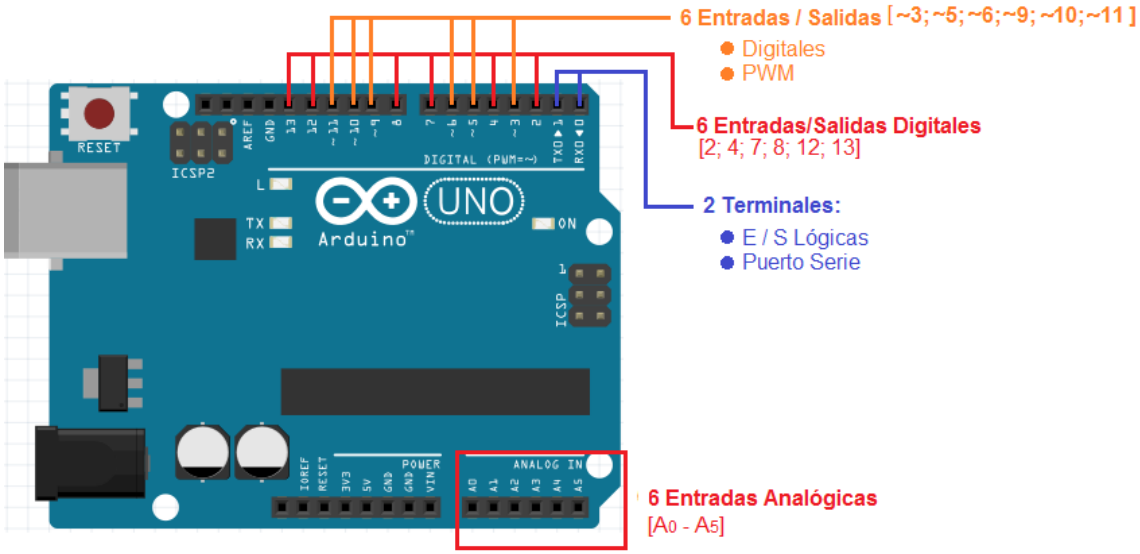


Ilustración 37. Placa - Arduino Pines utilizados

1. Un conjunto de 6 terminales que son entradas denominadas “Analógicas” porque cada una de ellas convierte la tensión que recibe entre 0V y 5V en un número comprendido entre 0 y 1024 aplicando una conversión analógico – Digital.
2. Un conjunto de 6 terminales que pueden ser definidas como entradas o salidas, y las denominamos “digitales”, porque sólo pueden tomar valores lógicos (“0”, “1” /LOW, HIGH) que se corresponden con 0 vol. y 5 vol.
3. Un conjunto de 6 terminales que pueden ser definidos como:
    a. Entradas o salidas “digitales”
    b. Salidas dando señales de pulsos modulados en duración (PWM)
4. Dos terminales que pueden ser definidos como:
    a. Entradas/Salidas lógicas
    b. Puerto Serie (Rx/Tx)

La configuración adoptada en el controlador es la que se muestra a continuación:

ARDUINO NODO

PIN	TIPO	Nombre	Funcionalidad
A0	E	Analog 0	Joystick (derecha + izquierda)
A1	E	Analog 1	Emergencia
A2	E	Analog 2	Luces de freno
A3	E	Analog 3	Batería
A4	E	Analog 4	
A5	E	Analog 5	
0	S	Rx	[Tx] Xbee
1	E	Tx	[Rx] Xbee
2	S	D0	SONIDO ** Futuro
3	S	Pwm0	
4	S	D1	Sincronismo
5	S	Pwm1	
6	S	Pwm2	
7	S	D2	Izquierda activa
8	S	D3	Derecha activa
9	S	Pwm3	
10		Pwm4	
11		Pwm5	
12	S	D4	Batería baja
13		D5	

Esquema

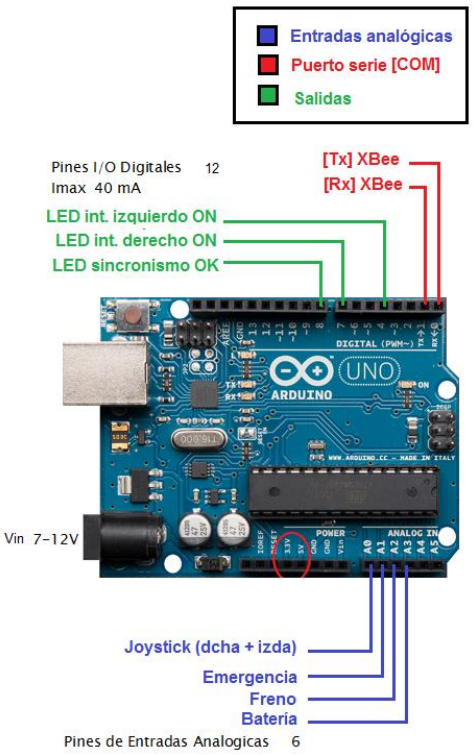


Ilustración 38. E / S NODO

ARDUINO COORDINADOR

PIN	TIPO	Nombre	Funcionalidad
A0	E		
A1	E		
A2	E		
A3	E		Batería
A4	E		
A5	E		
0	S	Rx	[Tx] Xbee
1	E	Tx	[Rx] Xbee
2	S	D0	
3	S	Pwm0	Int.Izquierdo
4	S	D1	
5	S	Pwm1	Int.Derecho
6	S	Pwm2	Freno
7	S	D2	
8	S	D3	
9	S	Pwm3	
10		Pwm4	
11		Pwm5	
12	S	D4	
13		D5	

Esquema

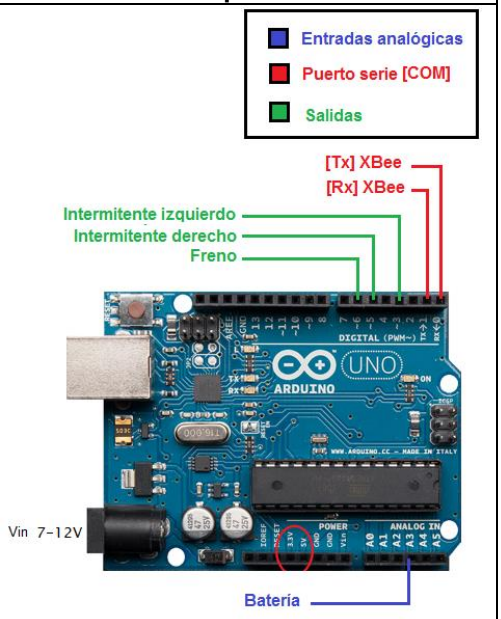


Ilustración 39. E/S COORDINADOR

Tabla 17. Arduino: Configuración de entradas / salidas.

5.2.2.1 ENTRADAS ANALÓGICAS

Contiene 6 entradas que acceden a convertidores analógicos digitales de 10 bits, lo que nos permite discriminar entre 2^{10} (1024) valores distintos.

A la hora de realizar la programación de estos pines, es importante tener en cuenta que como son sólo de entrada no es necesario declararlos como INPUT u OUTPUT en nuestro programa.

El controlador las distingue numerándolas del "0" al "5" [**A_0; A_1; A_2; A_3; A_4 y A_5**].

Las utilizamos:

- *En el NODO para la entrada de los pulsadores y para conocer el nivel de batería. [A_0; A_1; A_2; A_3]*
- *En el COORDINADOR para conocer el nivel de batería. [A_3]*

5.2.2.2 I/O DIGITALES

Si bien el modelo utilizado Arduino UNO nos ofrece 14 pines que pueden configurarse como Entradas o Salidas digitales, nosotros lo limitaremos a las 6 que el controlador distingue con la numeración: "{**2, 4, 7, 8, 12, 13**}", ya que el resto de pines pueden configurarse para que se comporten como PWM o comunicación puerto serie, lo cual aporta muchas ventajas/facilidades tanto para aplicaciones en este proyecto (comunicación inalámbrica Zigbee) como para futuros desarrollos.

Por lo tanto, en nuestro caso, el controlador nos ofrece 6 pines que:

- Podemos configurar para que funcionen tanto como entrada como salida digital, (funcionamiento no simultáneo). Es necesario identificarlo previamente en el código de nuestro programa.
- Los valores de alimentación habituales son 0V – 5 V, y una tensión umbral muy cercana a 2,5 V. Con nivel bajo de "0V.=LOW", y con nivel alto de "5 V.=HIGH". Para evitar daños en la placa, es importante no introducir valores mayores a 5V. En caso de que lo fueran, bastaría con introducir un circuito divisor de tensión.

  En el punto "***5.2.5.1*** *Circuito de entrada al Arduino nodo***"** realizamos un cálculo para las entradas de nuestros interruptores.

*En nuestro caso, hemos configurado estos pines como **salidas**:*

- *En el NODO: para los LEDs que indican el estado del funcionamiento del sistema*

5.2.2.2.1 I/O Digitales con funciones especiales

Salidas PWM .....I/O Digitales

Por último el controlador nos ofrece 6 salidas, con la numeración {3, 5, 6, 9, 10, 11}, con una señal que tiene la forma de un pulso modulado en amplitud (PWM), con una amplitud de "0V." a "5V." y un ancho del pulso comprendido entre los valores de "0" (cte=0V.) y "255" (cte=5V.). En la figura se muestra la señal

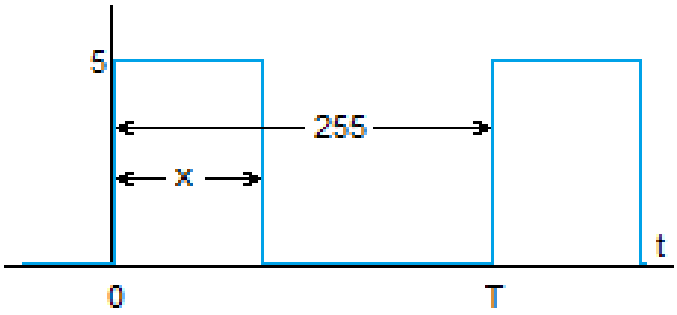


Ilustración 40. Señal PWM

Entre los diferentes usos que podemos darle a las salidas PWM se encuentra la de atenuar un LED (modulando el ancho de pulso), que en nuestro caso, utilizamos para mejorar el rendimiento y el consumo de nuestras señales en el ***módulo COORDINADOR****.*

Puerto Serie .....I/O Digitales

El puerto serie es una de las principales ventajas de la placa Arduino, ya que gracias a él comunicamos la placa con el ordenador, controlamos un robot o conectamos dispositivos como el XBee.

Para poder habilitar el puerto serie (envío de información mediante una secuencia de bits), es necesario disponer al menos de dos conectores para la comunicación: Receptor (Rx) y Transmisor (Tx).

El funcionamiento básicamente es el siguiente: el flujo de información llega y sale por el puerto serie del procesador Arduino. Cuando llega es almacenada en un buffer, para ser extraída y procesada por el programa que corre en el procesador.

5.2.2.3 ALIMENTACIÓN

Cuando hablamos de alimentación en Arduino UNO podemos referirnos a:

a) **Entrada de alimentación:** Alimentación requerida por la propia placa del controlador:

   a. A través de la conexión USB existente la placa se alimenta a 5V: es la que utilizamos durante la realización del proyecto para conectar la placa al ordenador y realizar las programaciones y pruebas necesarias.

b. Conectando la placa a una pila / batería externa. En este caso, se requiere una tensión comprendida entre 7 V y 12 V: Una vez finalizada la fase de pruebas, es necesario dotar de movilidad a nuestro sistema, y por tanto, como ya indicamos en el apartado "*4.5 Módulo de Alimentación*" hemos optado por baterías recargables con entrada estándar similares a las que se utilizan en los dispositivos móviles como tabletas o teléfonos (Baterías LiPo, con una vida útil larga, y distintos voltajes y capacidades).

Es importante tener en cuenta que todas las placas de Arduino tienen un regulador interno de tensión que precisa de un voltaje de entrada de aproximadamente 7V y que convierte **V_{in} a 5V** (y en algunos casos a 3,3V), por lo que sobretensiones supondrían un mayor consumo y sobrecalentamiento de nuestra placa y una alimentación superior a los 12V especificados podría dañarla.

b) **Salida de alimentación:** Pines de alimentación.

Según las especificaciones de la placa Arduino Uno[22],

El controlador proporciona la referencia de toma de tierra (GND) y salidas de alimentación: dos reguladas (5V y 3,3V) y otra Vin que entrega directamente la tensión de alimentación de la placa.

NODO			Esquema
PIN	**TIPO**	**Funcionalidad**	
IO_{ref}	S		
Reset	E		
3,3V	S	Alimentación XBee	
5V	S	Alimentación interruptores Izda/Dcha	
GND	S	Toma de tierra	
GND	S		
V_{in}	S	Tensión de la batería	
COORDINADOR			
PIN	**TIPO**	**Funcionalidad**	
IO_{ref}	S		
Reset	E		
3,3V	S	Alimentación XBee	
5V	S		
GND	S	Toma de tierra	
GND	S		
V_{in}	S	Tensión de la batería	

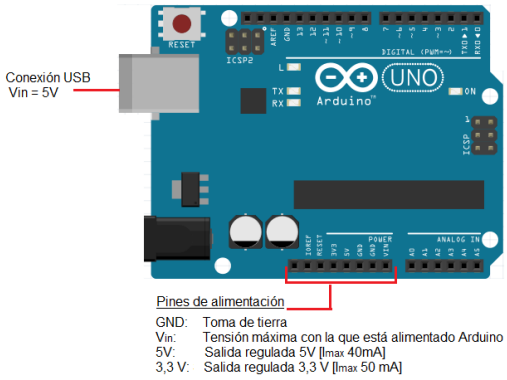


Tabla 18. Alimentación - Arduino

Utilizaremos la funcionalidad de comunicación por puerto serie, para el intercambio de información a través del dispositivo Radio (XBee).

Para ello, habilitamos como puerto serie los los pines **{0, 1}, {Rx, Tx},** con rangos de operación 0V – 5V.

[22] https://www.arduino.cc/en/Main/arduinoBoardUno

5.2.3 TRANSMISORES/RECEPTORES: XBEE

"ZigBee is the only open, global wireless standard to provide the foundation for the Internet of Things by enabling simple and smart objects to work together, improving comfort and efficiency in everyday life".

http://www.zigbee.org/

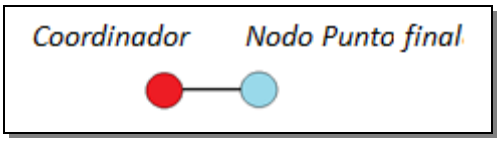
Los dispositivos empleados para la transmisión y recepción de la información son los módulos Xbee, que utilizan el protocolo Zigbee[23].

Cada módulo se identifica mediante una dirección de 64 bits que viene dada de fábrica, y cuando se asocia a una red Zigbee, el coordinador de la red le asigna una dirección de 16 bits[24].

Ilustración 41. XBee

En el punto "*5.1 Restricciones básicas.*" la "*Ilustación 31. Tipos de redes Xbee"* presentamos los tipos de redes y elementos que componen el sistema de comunicaciones.

El Sistema de señalización en bicicleta estará compuesto por un Coordinador y por un Dispositivo final o Nodo.



Coordinador.

Su función es formar una red. Para ello, es responsable de establecer el canal de comunicaciones y de fijar el identificador de red (PAN ID).

Una vez establecidos estos parámetros y formada la red, el Coordinador:

- Permite unirse a la red al resto de nodos.
- Actúa como origen y/o destinatario de información, tratándola y reenviándola si procede.

Nodo - Dispositivo Final.

El NODO o dispositivo final, precisa de la existencia de un COORDINADOR para el funcionamiento del sistema, ya que no puede enviar información directamente a otros dispositivos.

[23] Zigbee está basado en el estándar de comunicaciones para redes inalámbricas IEEE_802.15.4.
[24] El número máximo de direcciones de red que se pueden asignar, y por tanto de elementos en una red Zigbee es 2^{16} =65.535

5.2.3.1 ESQUEMA DE FUNCIONAMIENTO.

Utilizaremos los módulos XBee con sus correspondientes “Shields” para realizar una integración sencilla con Arduino.

Entendemos como shields a las placas impresas que se pueden conectar directamente en la placa Arduino para ampliar sus capacidades.

Si bien en los anexos del presente proyecto se puede encontrar una descripción más detallada[25] de XBee y sus funcionalidades, se incluye a continuación un esquema con los datos más relevantes.

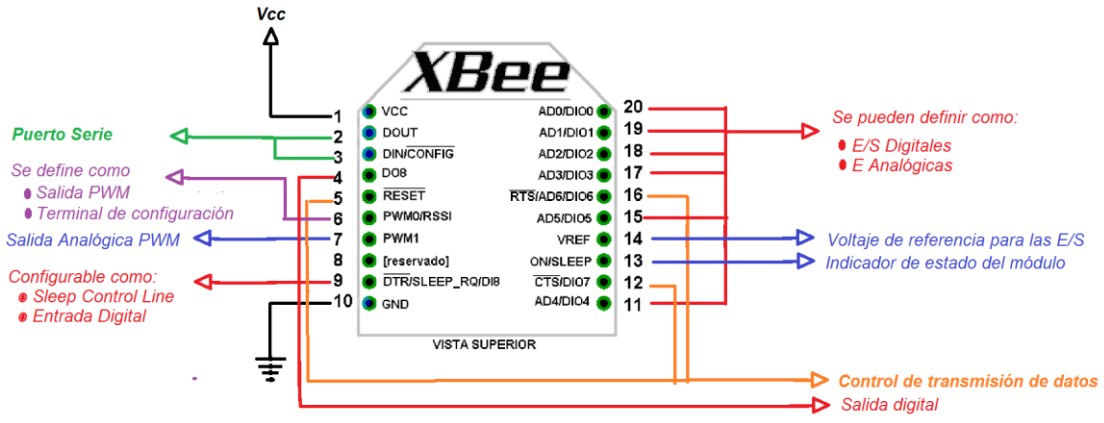
Utilizaremos módulos XBee de la serie 1
Existen 2 series: • Serie 1 (se utiliza en redes punto a punto - punto a multipunto). • Serie 2 (se utiliza en aplicaciones más complejas que requieren repetidores). Aunque ambos tienen el mismo pinout, no son compatibles entre sí (chipset y protocolos distintos)
Modo Transmisión Serie Transparente y comunicación bidireccional
El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TXD y RXD) para comunicarse con un microcontrolador, o directamente a un puerto serial utilizando algún conversor adecuado para los niveles de voltaje En este modo todo lo que ingresa por el pin 3 (Entrada de Datos), es guardado en el buffer de entrada (hasta 100 bytes) y luego transmitido. Y todo lo que ingresa como paquete RF, es guardado en el buffer de salida y luego enviado por el pin 2 (Salida de Datos).
Esquema básico
Usaremos en el proyecto los pines de alimentación (Vcc y GND) y los pines {2,3} para la comunicación puerto serie.    *Ilustración 42. Pines XBee[26]*

Tabla 19. Esquema funcionamiento XBee

[25] Información copleta de xbee en http://www.digi.com/lp/xbee/
[26] Imagen extraída de: https://hangar.org/webnou/wp-content/uploads/2012/01/Capsulab081.pdf

5.2.3.2 TIPO DE COMUNICACIÓN: MODO AT

La comunicación para nuestro proyecto es punto a punto, donde no es necesario ningún tipo de control y por lo tanto el modo AT es el sistema idóneo. Esta configuración, no permite el uso de Control de Flujo (RTS & CTS), por lo que ésta opción debe estar desactivada en el terminal y en el módulo XBEE.

El NODO y el COORDINADOR pueden configurarse para comunicarse en dos modos diferentes:

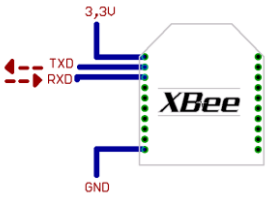
Modo AT - Transmisión Serie Transparente.	**Modo API**
Envío de datos por puerto serie usando los pines Rx/Tx. Es el método más sencillo y al contar solo con dos elementos **nuestro sistema no se ve afectado** por las limitaciones de configuración en el caso de sistemas con múltiples elementos (Mallas) Por lo tanto, no es necesario emplear el modo API (más complejo y requiere más recursos)	Es un modo más complicado que requiere de programación específica para poder dotar de flexibilidad al sistema. En función de la secuencia de bits recibida los diferentes módulos interpretarán qué tipo de operación deben realizar, es decir, a qué función deben llamar.
 Ilustración 43. Conexiones mínimas Rx/Tx Xbee[27]	 Ilustración 44. Estructura de trama.[28]
EL QUE UTILIZAREMOS.	NO LO UTILIZAMOS

Tabla 20. Modo AT vs Modo API

DESARROLLOS FUTUROS: Un controlador – Diversos nodos.
Una de las ideas de desarrollo futuro, es diseñar un sistema en la que desde un dispositivo COORDINADOR se controlen simultáneamente multitud de NODOS.
Una aplicación directa de esta funcionalidad es, por ejemplo, la circulación de grupos:

- *Visitas guiadas en bicicleta, segway, etc. En las que el guía realiza la pulsación y la señal de giro se activa en todos los vehículos.*

[27] Fuente de imagen: https://hangar.org/webnou/wp-content/uploads/2012/01/Capsulab081.pdf
[28] Fuente de imagen: http://fuenteabierta.teubi.co/2014/03/arduino-y-el-xbee-series-1-modo-api.html

5.2.4 SEÑALIZACIÓN: LED

Para el desarrollo del proyecto se han elegido led en tiras RGB. Esta elección ha sido realizada por la posibilidad de modificar los colores y utilizar los mismos componentes para generar luz roja, ámbar o blanca…

Lo primero que hemos hecho ha sido medir el consumo de cada uno de los colores básicos. El banco de medida empleado para la medida es el indicado en la figura:

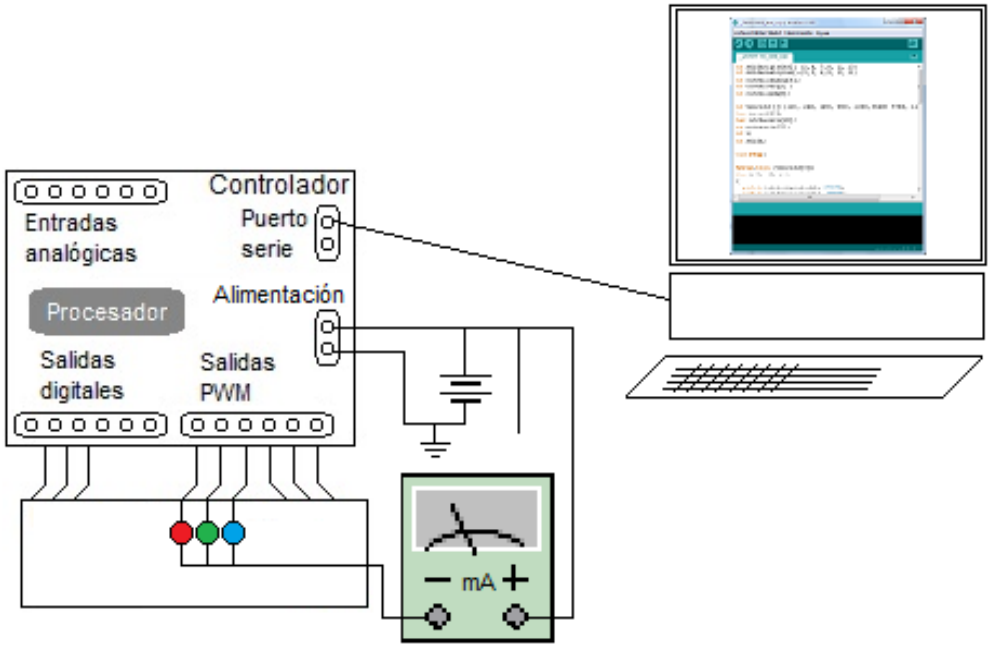


Ilustración 45. Banco de medida Luminosidad – Corriente del LED

Para controlar la corriente que pasa por los LEDs se ha programado[29] el controlador para ir variando las salidas de pwd entre 0 y 255, que corresponde a una salida todo el tiempo en 0V y una salida todo el tiempo en 1V.

A continuación se muestran los resultados obtenidos en función del color analizado.

[29] Ver Anexos - Programas

Consumo vs Intensidad - RGB

Se ha programado para que la salida PWM del controlador a la que está conectada la entrada de cada color (Rojo, Verde y Azul) del led RGB, de saltos de 15 unidades (0-255) cada cuatro segundos.

Rojo				
Vpwm	Irojo (mA)	□ Irojo (mA)	Percepción	
0	0	0		
15	0,93	0,93	0,9	
30	1,88	0,95	1,8	
45	2,8	0,92	2,7	
60	3,76	0,96	3,6	
75	4,7	0,94	4,5	
90	5,6	0,9	5,5	
105	6,6	1	6,4	
120	7,55	0,95	7,3	
135	8,5	0,95	8,2	
150	9,45	0,95	8,9	
165	10,39	0,94	9,4	
180	11,37	0,98	9,6	
195	12,29	0,92	9,8	
210	13,24	0,95	9,9	
225	14,2	0,96	10,0	
240	15,15	0,95	10,0	
255	16,1	0,95	10,0	

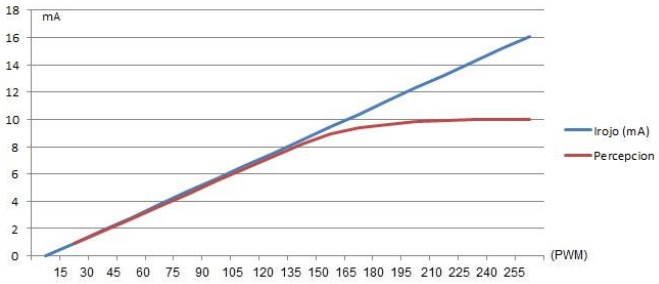


Ilustración 46. Consumo vs intensidad LED rojo

En la figura se muestra la variación de la corriente según el incremento de ancho de pulso y la percepción subjetiva del cambio de intensidad.

Verde				
Vpwm	Irojo (mA)	□ Irojo (mA)	Percepción	
0	0	0	0	
15	0,59	0,59	1	
30	1,18	0,59	2	
45	1,78	0,6	3	
60	2,38	0,6	4	
75	3	0,62	5	
90	3,6	0,6	6	
105	4,23	0,63	7	
120	4,85	0,62	8	
135	5,48	0,63	8,2	
150	6,11	0,63	8,7	
165	6,75	0,64	9,2	
180	7,4	0,65	9,6	
195	8,05	0,65	9,8	
210	8,7	0,65	9,9	
225	9,36	0,66	9,95	
240	10,02	0,66	10	
255	10,67	0,65	10	

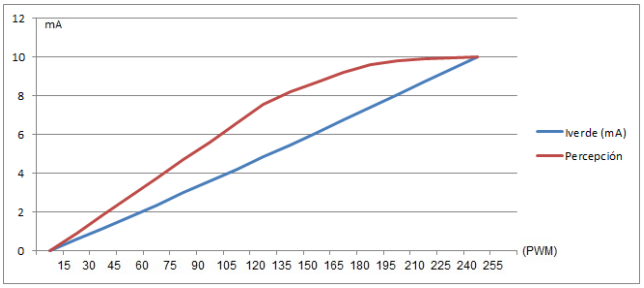


Ilustración 47. Consumo vs intensidad LED verde

En la figura se muestra la variación de la corriente según el incremento de ancho de pulso y la percepción subjetiva del cambio de intensidad.

Azul				
Vpwm	Irojo (mA)	□ Irojo (mA)	Percepción	
0	0	0	0	
15	0,56	0,56	0,95	
30	1,18	0,62	1,9	
45	1,7	0,52	2,8	
60	2,28	0,58	3,7	
75	2,86	0,58	4,6	
90	3,45	0,59	5,5	
105	4,04	0,59	6,4	
120	4,62	0,58	7,3	
135	5,22	0,6	8,2	
150	5,82	0,6	9	
165	6,4	0,58	9,7	
180	7,03	0,63	9,83	
195	7,64	0,61	9,95	
210	8,25	0,61	9,97	
225	8,87	0,62	9,99	
240	9,49	0,62	10	
255	10,12	0,63	10	

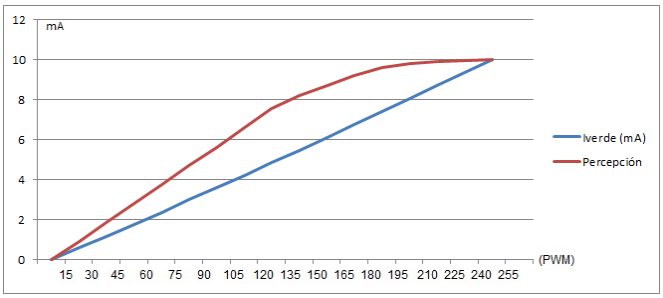


Ilustración 48. Consumo vs intensidad LED Azul

En la figura se muestra la variación de la corriente según el incremento de ancho de pulso y la percepción subjetiva del cambio de intensidad.

Tabla 21. Consumo vs intensidad LEDs rojo, verde y azul.

5.2.5 DESARROLLOS ESPECÍFICOS:

El intercambio de señales entre los distintos elementos del sistema no siempre se realiza de una forma directa.

Las señales proporcionadas por los sensores hay que adaptarlas a los niveles que requieren las entradas a los módulos controladores, es decir, las placas Arduino.

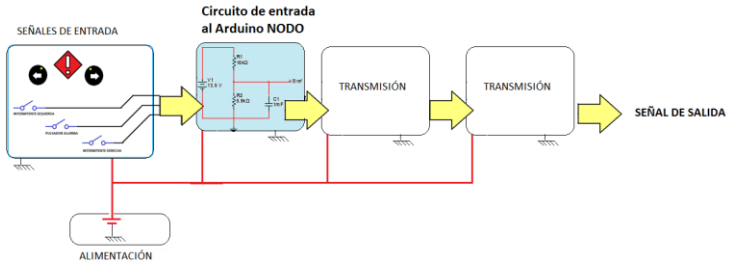


Ilustración 49. Acondicionamiento de señal - NODO

A su vez, las señales que proporcionan los módulos controladores, tienen que aportar la energía suficiente para las entradas de los módulos de luces.

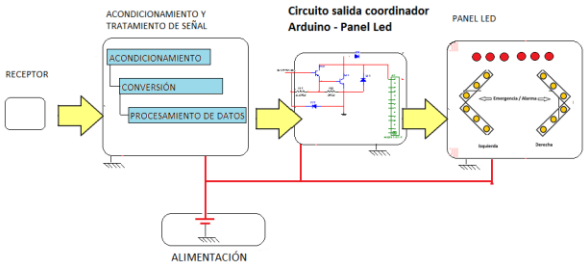


Ilustración 50. Acondicionamiento de señal - COORDINADOR

5.2.5.1 CIRCUITO DE ENTRADA AL ARDUINO NODO

Los sensores entregan una información que hay que adaptarla a las tensiones que aceptan las entradas del controlador. La batería tiene una tensión nominal de doce voltios, pero suele estar en un rango entre 13,8V. y 10,8V con un funcionamiento correcto (12V.). Sin embargo los niveles de tensión que admiten las entradas analógicas están entre 0-5V.

Así para monitorizar la tensión de la batería utilizaremos un divisor resistivo, de tal forma que para 13,8V de entrada corresponda una tensión de salida de 5V. Para una tensión 10,8 voltios, corresponderá una tensión de salida de 3,9V, por debajo de la cual tendremos que disparar una alarma de batería baja.

El circuito es el mostrado en la figura:

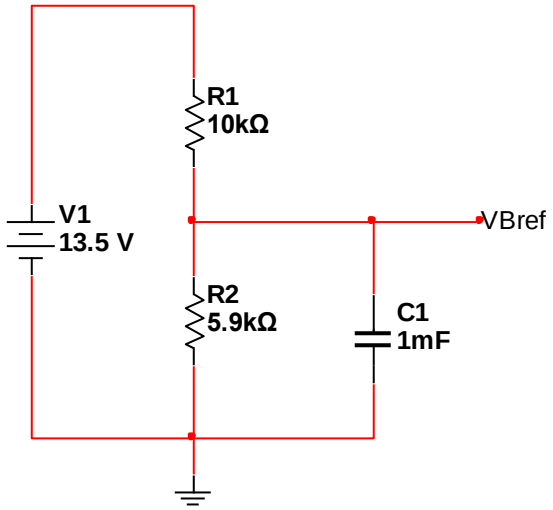


Ilustración 51. Esquema detección de tensión de la Batería

El condensador evitará que entre ruido que pueda activar indebidamente la alarma.

5.2.5.2 CIRCUITO DE SALIDA ARDUINO CONTROLADOR – LED.

En el cuadro de la figura se da las características técnicas del módulo controlador

Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada	7 – 12V
Voltaje de entrada (Límite)	6 – 20V
Pines entrada-salida digital.	14 (6 pueden usarse como salida de PWM)
Pines entrada analógica.	6
CC por pin IO	40 mA
CC 3.3V	50 mA
Memoria Flash	32 KB (0,5 KB ocupados por el bootloader)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Tabla 22 Características técnicas Arduino

La corriente máxima que circula por cada tres led es 14mA. Si se van a poner un conjunto de 6 led para cada intermitente, tendremos una corriente de entrada de 24mA, lo que supone una potencia consumida de 290 mWatt cuando está encendido un intermitente o la luz de freno.

La Corriente máxima por pin es de 40 mA. Por lo tanto hemos incluido un driver que puede aguantar una corriente máxima por pin de 500mA.

Para cada una de las salidas incluiremos un driver como el de la figura:

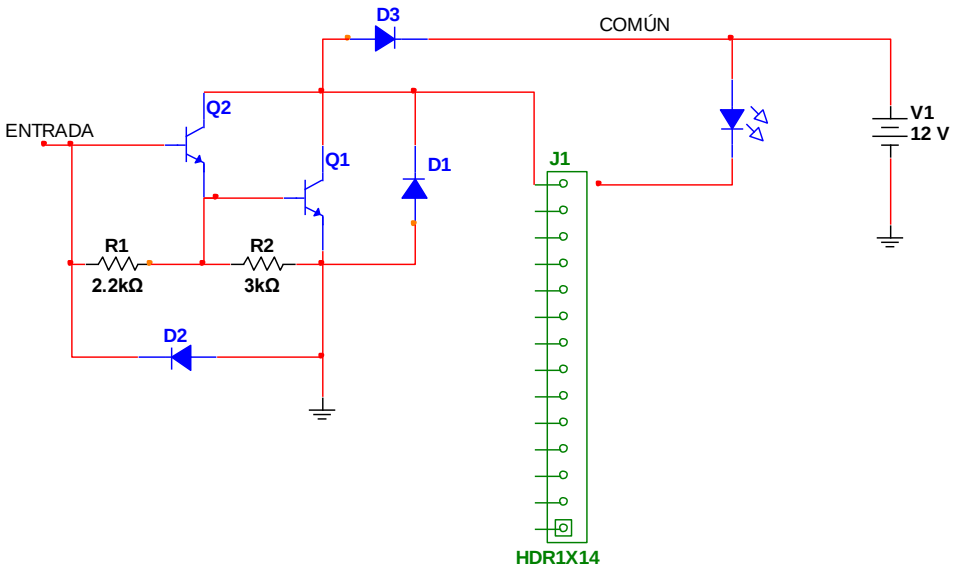


Ilustración 52. Buffer para control del encendido de un Led

Utilizaremos el integrado uLn2003 como driver de salida del controlador de cara al sistema de señalización. El esquema de dicho semiconductor es el mostrado en la figura:

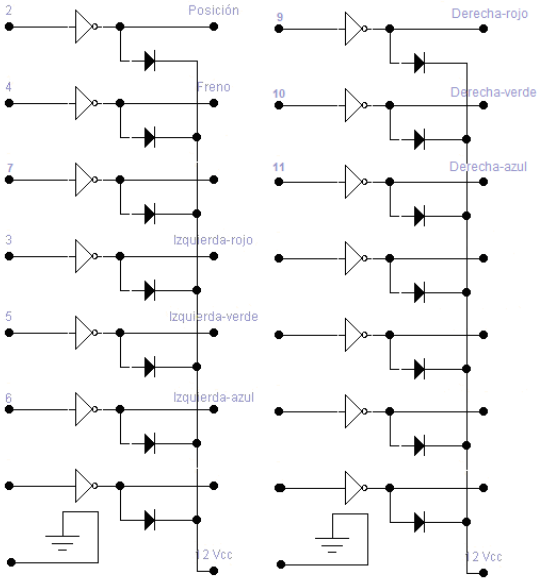


Ilustración 53. Conexión de los uLn2003 utilizados

El esquema de conexión empleado entre el circuito de salida del controlador del módulo Coordinador y el sistema de señalización Led empleado en el proyecto es el mostrado en la figura.

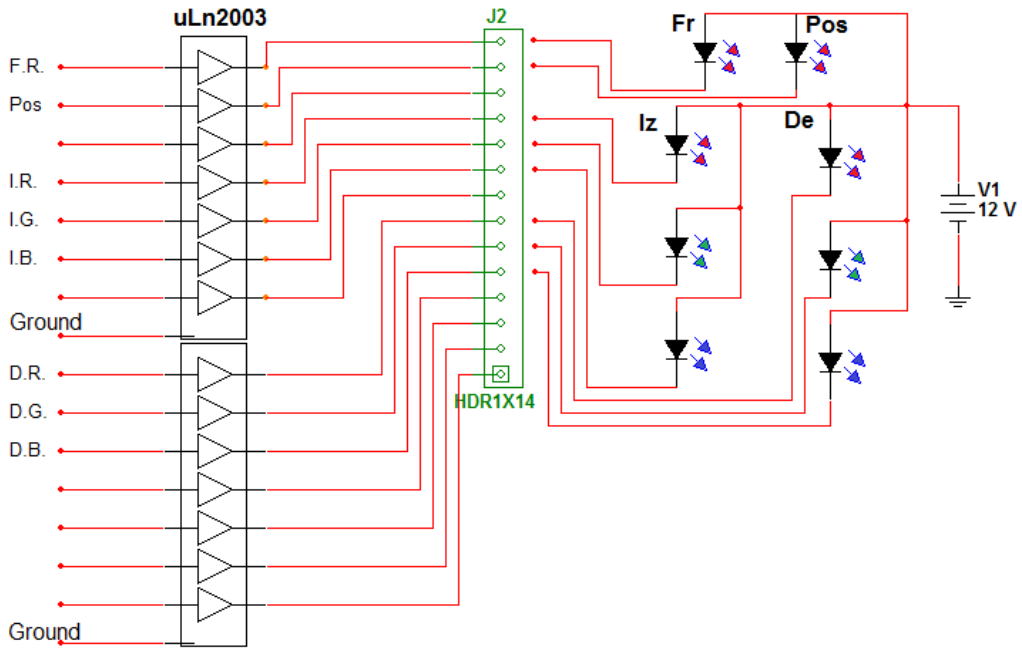


Ilustración 54. Conexión Coordinador – Panel de señalización LED

5.3 SOFTWARE

En este punto nos centramos en la programación y el software requerido para el funcionamiento de nuestro sistema.

Para ello se ha realizado un conjunto de configuraciones y rutinas para la detección, autenticación, transmisión y procesado de la información que engloba a los módulos de control y transmisión (Arduino y XBee).

Ambos módulos cuentan con aplicaciones propias para facilitar la configuración y el desarrollo de nuevas funcionalidades, y en nuestro caso utilizamos la versión para SO Windows.

Módulo	**Interfaz – Software**	**Lenguaje**	**Tareas**
Control Arduino		Processing (similar a C++)	Configuraciones Programas específicos
Transmisión XBee		X-CTU Digi	Configuraciones

Tabla 23. Resumen tecnología - Lenguaje - Interfaz

Durante el desarrollo de esta memoria se realizarán esquemas y diagramas de flujo de los programas desarrollados incluyéndose la programación completa en Anexo correspondiente a los planos y rutinas.

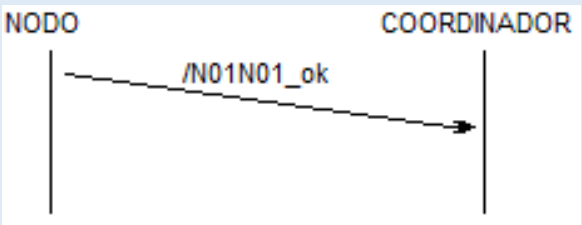
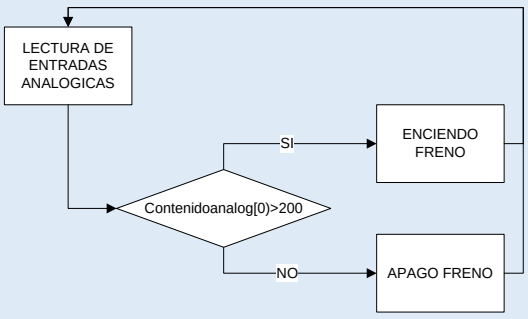
Esquema	**Diagrama de flujo**
	

Tabla 24. Muestra de Esquema - Flujograma

5.3.1 TRANSMISORES / RECEPTORES: XBEE

*Inicialmente consideramos realizar una programación como "**cable virtual**", en el que cualquier dato en los pines de un módulo XBee se traslade automáticamente al mismo pin del módulo asociado.*
*Sin embargo, la configuración más adecuada a nuestras necesidades es la que se conoce como "**Conexión transparente**", tipología punto a punto.*

Como hemos indicado anteriormente, para la transmisión de la información hemos usado los módulos XBee y para su configuración utilizamos la herramienta XCTU suministrada por el propio fabricante.

Para facilitar su identificación, se nombra Xbee-Coordinador al dispositivo que se encuentra conectado físicamente al Arduino Coordinador (parte trasera del dispositivo) y Xbee-Nodo al que se conecta al Arduino Nodo (manillar del ciclista).

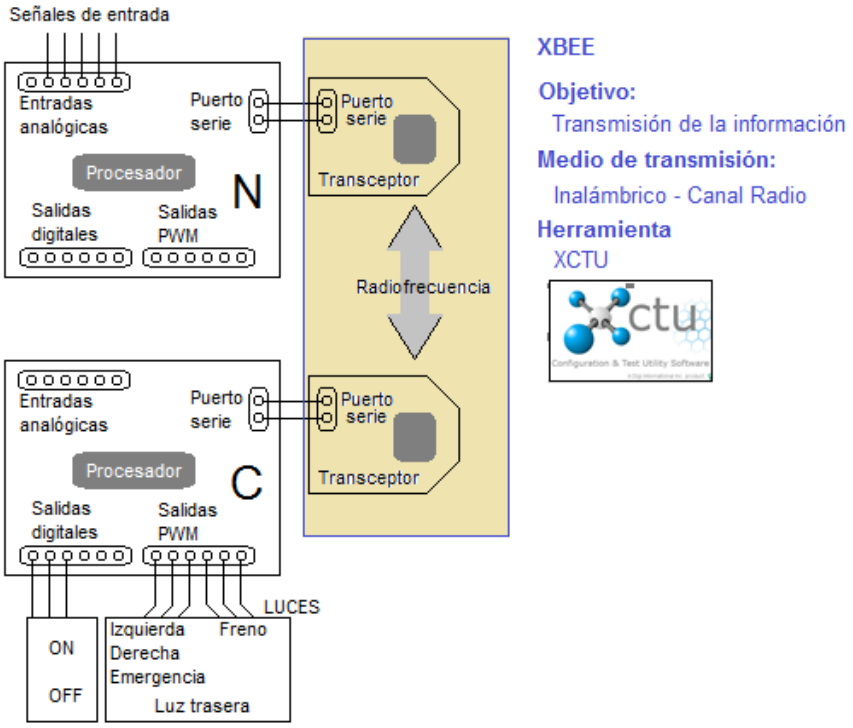


Ilustración 55. Identificación de componentes - Xbee

La configuración de los módulos se puede realizar tanto por comandos AT[30] (pueden ser controlados mediante programas a través del puerto serie del dispositivo, y es utilizado en aplicaciones de integración como por ejemplo mediante instrucciones desde Arduino) como por la herramienta XCTU (Ordenador).

[30] En nuestro caso utilizamos la herramienta XCTU. Para ampliar información de programación por comandos, ver http://www.digi.com/lp/xbee/

5.3.1.1 INTERFAZ: XCTU

Es el software desarrollado por Digi y disponible para Windows (en pruebas para otros S.O.) que proporciona un interfaz amigable para la configuración de los módulos XBee.

Dado que en nuestro proyecto la interconexión funciona como una "Conexión transparente", nos limitaremos a utilizar XCTU para la configuración de los parámetros requeridos.

Introducción: Acceso e Interfaz

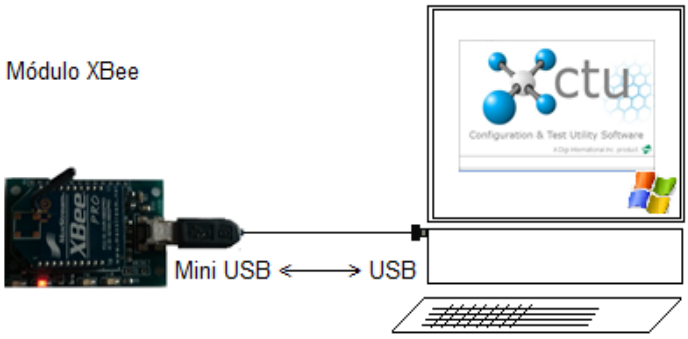


Ilustración 56. Conexión XBee - PC

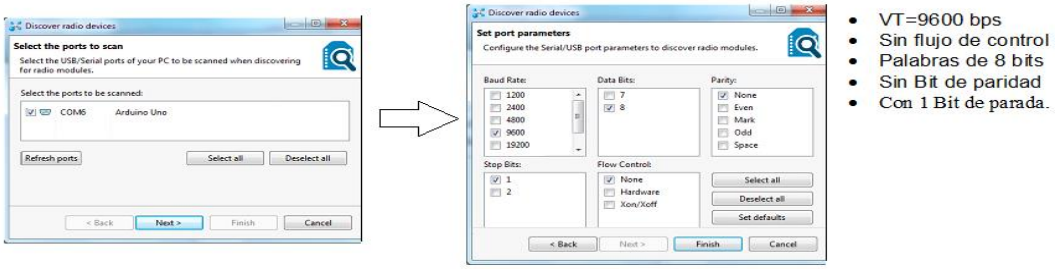
Podemos conectar uno o varios módulos al PC mediante cables Mini USB <--> USB a nuestro equipo.

Para ello se utiliza una placa específica de adaptación que nos proporciona el conector adecuado. El interfaz utilizado es el XCTU versión 6.1.0.

Configuración de parámetros[31]

Aunque pueden conectarse inicialmente de forma simultánea ambos dispositivos, en nuestro caso primero conectamos y configuramos el coordinador (opción de búsqueda de dispositivos "") y posteriormente, mediante la opción de agregar nuevo dispositivo "", agregamos el nodo.

1. Conectamos el Coordinador y configuramos los parámetros de comunicación ordenador – dispositivo.



- VT=9600 bps
- Sin flujo de control
- Palabras de 8 bits
- Sin Bit de paridad
- Con 1 Bit de parada.

Ilustración 57. XCTU – Configuración de parámetros 1

[31] La configuración realizada están en la Tabla 25. Parámetros de configuración como Conexión Transparente.

Para que se pueda transferir información entre el programa gestor del puerto serie y el módulo Xbee, ambos tienen que tener configurados los mismos valores de velocidad y estructura de la información.

2. En el panel de la aplicación "Radio Modules" aparece una imagen por cada dispositivo conectado, y en "Radio Configuration[32]" las opciones de configuración. Establecemos los valores de los parámetros de comunicación radio del Coordinador para su funcionamiento como "conexión transparente".

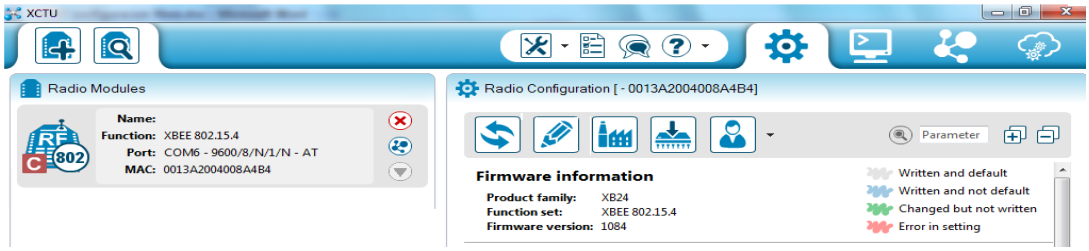


Ilustración 58. XCTU – Configuración de parámetros 2

3. Conectamos el segundo módulo como nodo. Configuramos tanto los parámetros de comunicación ordenador – dispositivo como los parámetros de comunicación radio.

Ilustración 59. XCTU – Configuración de parámetros 3

En la "Tabla 25. Parámetros de configuración como Conexión Transparente." adjunta, mostramos los parámetros que se han modificado y sus valores correspondientes.

Se han elegido las siguientes condiciones de funcionamiento:

- Que el Canal de transmisión que sea el predefinido como "C"
- Configurar una Red de Área Local Privada (PAN) manteniendo el valor del ID que viene por defecto.
- Que la dirección del módulo Coordinador sea la 612 y la del Nodo la 613.
- Que los módulos Xbee manden los datos que componen la información tan pronto como lleguen a los buffer de entrada.
- Que ambos módulos controlen el flujo de información transmitida.

[32] La configuración se puede hacer de una forma dirigida en "Radio Configuration" o por comandos.

TABLA DE CONFIGURACION "CONEXIÓN TRANSPARENTE" DE DISPOSITIVOS

	Xbee Coordinador	***Xbee Nodo***	***Descripción***
Networking & Security			
CH *Channel*	C	C	Establece el canal por el cual se realiza la conexión RF entre módulos. **Usamos el canal C**.
ID *PAN ID*	3332	3332	ID de red privada 3332
CE *Coordination Enable*	1	0	Comportamiento del módulo. 0 = Nodo 1 = Coordinador
MY *16 bits Source Addr*	612	613	Dirección propia (origen)
DL *Destination Address Low*	613	612	Direccionamiento: Ajuste de 32 bits menos significativos
DH *Destination Address High*	0	0	Direccionamiento: Ajuste de 32 bits más significativos
A1 *End Device Association*		0	PAN ID Coordinador = PAN ID Nodo CH Coordinador = CH Nodo
A2 *Coordinator Association*	0		PAN ID Estático. Opera en canal fijo dado por CH.
RF Interfacing			
PL *Power Level*	Lowest	Lowest	Potencia de emisión radio
Sleep Modes			
Serial Interfacing			
BD *Interface Data Rate*	6	6	Tasa de transmisión 57600
RO *Packetization timeout*	0	0	Envío de paquetes según se reciben por el puerto serie.
I/O Settings			
Diagnostics			
A/T Command Options			

Tabla 25. Parámetros de configuración como Conexión Transparente.

5.3.2 MÓDULO DE CONTROL: ARDUINO.

En este apartado describiremos el software empleado en la programación e incluiremos otros elementos, como diagramas de flujo, que recojan de una forma sencilla los comportamientos y rutinas que gestionan el sistema.

En algunos casos, se incluirá una descripción de instrucciones utilizadas en los programas[33] que están disponibles en Anexos.

El funcionamiento global que queremos programar es el que se muestra en la Ilustración 60. En ella identificamos diferentes comportamientos en función de si la autenticación ha sido correcta, si es necesaria la transmisión de información y de si debe actuar el procesador del Nodo o del Coordinador.

Desglosaremos cada caso particular en los apartados siguientes.

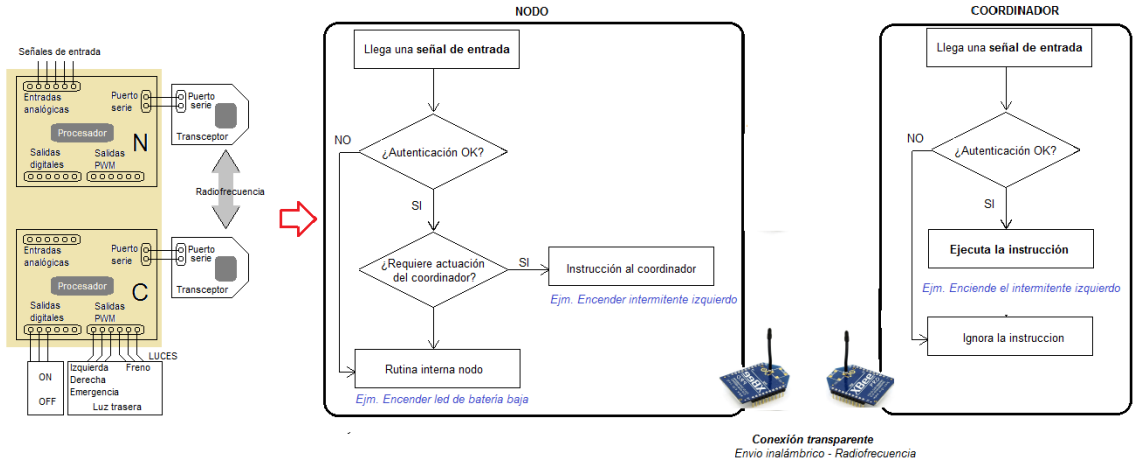


Ilustración 60. Identificación de componentes – Arduino.

Indicar que de entre las múltiples soluciones de programación, se ha optado por aquellas que he considerado más didácticas, a pesar de que pudieran no ser las más eficientes.

[33] Los programas completos se incluyen en el Anexo correspondiente.

5.3.2.1 INTERFAZ Y ESTRUCTURA DE PROGRAMACIÓN

La programación de los módulos de control se hace mediante un interfaz [34]que facilita el fabricante de los controladores a través del enlace http://arduino.cc/en/Main/Software .

Módulo	Interfaz – Software	Lenguaje	Tareas
Control Arduino		Processing (similar a C++)	Configuraciones Programas específicos

Dado que el modelo de placa que utilizamos es Arduino UNO, podemos conectar nuestro hardware al ordenador directamente mediante un cable USB.

Para comenzar accedemos al entorno de desarrollo y establecemos la comunicación PC – Arduino seleccionando el tipo de placa y el puerto COM al que está conectado desde el menú de herramientas.

A la hora de realizar los programas, es importante tener en cuenta la estructura de programación, según se muestra en la *Ilustración 61. Estructura básica programa Arduino.*

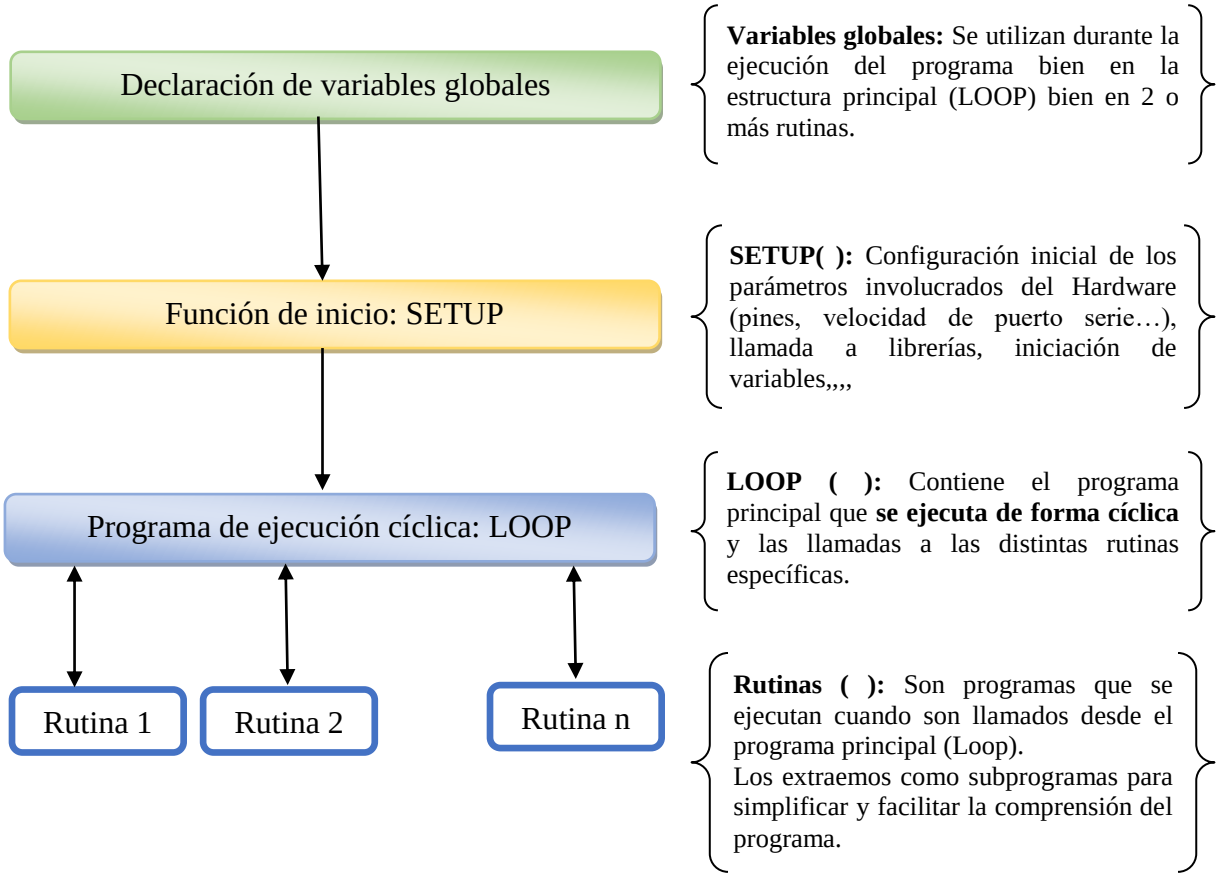


Ilustración 61. Estructura básica programa Arduino

5.3.2.2 CONSIDERACIONES GENERALES

Los puntos que se describen a continuación, deben ser tenidos en cuenta en la programación de ambos módulos (Nodo y Coordinador):

- Las entradas analógicas admiten un rango de [0v – 5V] y son leídas por el programa con un número comprendido entre 0 y 1023 aunque su estado sea pulsación ON – OFF, porque entran por pines analógicos.

- Para las salidas, asociadas a pines digitales, los valores programados son 0V o 5V.

- La alimentación de ambos módulos será mediante baterías de 9V. Se considera batería baja cuando su valor desciende a 5V.

- **Intercambio de órdenes entre módulos.**

  La presencia o ausencia de señales en las entradas analógicas del controlador del NODO, genera intercambio de información entre controladores.

  - Sólo se informa de los cambios de estado en las entradas: No se envía una orden que ya se está ejecutando.

    Ejemplo, sólo se manda la orden de activar freno al coordinador, cuando la variable que contiene la información (contenidoanalog[0]) cambia a un valor de uno. Mientras permanece activada no hay intercambio de información entre módulos.

  En la tabla siguiente se muestra la codificación elegida para esta información:

Cod.			**Significado**	**Orden**
F	**r**	**X**	Freno	"*x*=1" encender freno. "*x*=0" apagar freno.
I	**z**	**X**	Intermitente Izquierdo	"*x*=1" encender intermitente izquierdo. "*x*=0" apagar intermitente izquierdo.
D	**e**	**X**	Intermitente Derecho	"*x*=1" encender intermitente derecho. "*x*=0" apagar intermitente derecho.
E	**m**	**X**	Emergencia	"*x*=1" encender intermitentes izquierdo y derecho. "*x*=0" apaga los intermitentes derecho e izquierdo.
N	**b**	**X**	Respuesta estado de "batería"	"*x*=1" nivel correcto de batería "*x*=0" nivel de batería que no es correcto.
S	**i**	**X**	Solicitud de sincronismo	"*x*=1" petición de sincronismo. "*x*=0" error en sincronismo.
B	**a**	**X**	Estado de la batería del módulo nodo	"*x*=1" Nivel de batería ok. "*x*=0" Nivel de batería bajo.
B	a	c	Estado de la batería del módulo coordinador	----
L	a	t	Información de ""latido de sincronismo"	----
E	r	r	Se ha producido un error y hay que volver a sincronizarse.	----
N	0	1	Identificador del Nodo	----
C	0	1	Identificador del Coordinador	----

Tabla 26. Códigos de transferencia de órdenes

5.3.2.3 PUERTO SERIE: COMUNICACIÓN Y SINCRONISMO

Para el correcto funcionamiento del producto, es preciso que se establezca una comunicación entre el NODO y el COORDINADOR.

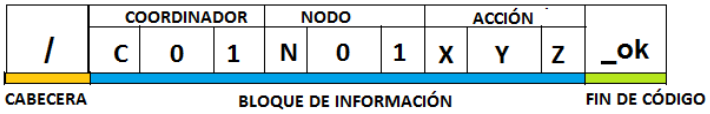
En nuestras placas Arduino, los datos serán enviados y recibidos a través de los pines 0 ($\mathbf{R_x}$) y 1 ($\mathbf{T_x}$) correspondientes al **puerto serie**.

Estos pines pueden funcionar bien como E/S Digitales, bien como puerto serie, por lo que es necesario definir su comportamiento al inicializar nuestro programa. Para ello utilizamos la instrucción "*Serial.begin ()*"

Para enviar los datos utilizaremos "*Serial.print ()*".

En cuanto a la recepción, creamos un buffer en el que almacenamos toda la información que llega, y utilizamos las instrucciones "*serial.available()*" y "*serial.read()*" principalmente. Así mismo, generamos un vector "***dato[ ]***" para almacenar los dígitos del bloque de información.

Para la comunicación por el puerto serie del controlador, se fijó[35] inicialmente como condición que las tramas debían tener la siguiente estructura: 5.1



PROCESSING: Ejemplo de envío de instrucción.

- Definimos los pines 0 y 1 como Rx y Tx:

  Serial.begin (9600) *// Define los pines 0 y 1 como Rx y Tx*

- Definimos el código global para la autenticación:

  *char codigo_global **[6]** = {"C","0","1","N","0","1"] // Define la variable código global*

- Tx: Transmitimos la orden:

  Serial.print("/"); *// Escribo la cabecera de la trama*

  Serial.print("código_global"); *// variable global: "C01N01"*

  Serial.print("orden"); *// Ejm FR1 → Enciende freno*

  Serial.print("_ok"); *// Fin de trama*

- Rx: Recibimos la orden:

  buffer = serial.available( ) *// Define un buffer*

  if (buffer>0) *// Si buffer > 0 es que ha llegado algo*

  { *dato[0] = serial.read()* *// cojo el primer dato recibido y lo meto en dato[0]*

  … }

  Comparando los datos recibidos, obtengo la cabecera, el fin de código y el bloque de información con las instrucciones a ejecutar.

[35] 5.2.2 Restricciones básicas.

Denominamos "***sincronización***" al proceso en el que ambos módulos se detectan y se fija el código que protegerá el intercambio de información.

Dada la importancia de que ambos módulos estén sincronizados, establecemos en nuestro código rutinas de "sincronización inicial" y "Comprobación de latido":

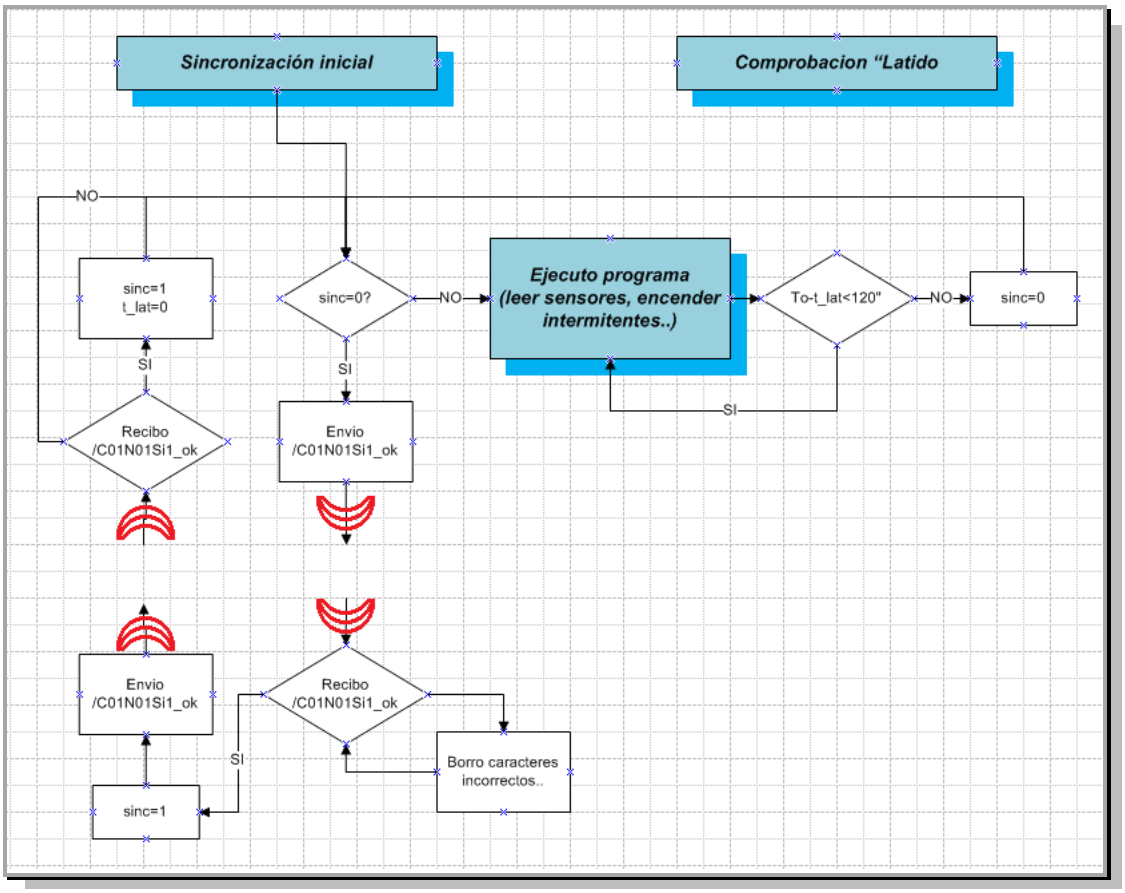


Ilustración 62. Sincronización NODO – COORDINADOR

- **Sincronización inicial** → Parte del programa mediante la cual se realiza la sincronización.

  1. Definimos las variables globales con los códigos del nodo y del coordinador:

     char código_global[6]={C,0,1,N, 0, 1};

  2. El nodo envía una solicitud de forma cíclica (t = 1 seg) → **/C01N01Si1_ok**

  3. El coordinador cuando recibe la señal del nodo y comprueba que es correcta, envía una señal de ok. → **/C01N01Si1_ok**

  4. El nodo recibe la señal, la verifica y continúa con el programa de lectura de datos y ejecución de órdenes.

- **Comprobación de "Latido"** → Sirve como mecanismo de revisión. Se ejecuta para verificar que la comunicación no se ha perdido durante el funcionamiento normal del sistema.

Adicionalmente, se ha incluido un led de señalización de sincronización en el manillar que pueda ser visible por el usuario cuyo funcionamiento se ha detallado en el punto "5.3.2.4.1 Lectura de entradas."

5.3.2.4 MÓDULO NODO

Este módulo se ha diseñado para **recibir y procesar** los datos que proceden:

- De los sensores (intermitentes, alarma, freno) .....{**A_0, A_1, A_2, A_3**}
- Del coordinador a través del radioenlace.....0 [**Rx**]

Por otro lado genera la señalización en los pines de **salida** para:

- Sincronizarse y enviar instrucciones al Coordinador.....1 [**Tx**]
- Monitorizar alarmas y estado de la señalización.....{4,7,8,12}

PIN	TIPO	Nombre	Funcionalidad	Definición variable Setup ( )
A0	E	Joystick (derecha + izquierda)	Entrada con información del sensor Joystick (derecha e izquierda)	**"Lectura_A0"**
A1	E	Emergencia	Entrada con información de pulsación señal de emergencia	**"Lectura_A1"**
A2	E	Luces de freno	Entrada con información de pulsación de frenado	**"Lectura_A2"**
A3	E	Batería	Entrada con información sobre el estado de carga de la batería	**"Lectura_A3"**
0	E	[Rx] Xbee	Se utiliza para la recepción de información a través del canal de transmisión	***Serial.begin*** **( )**
1	S	[Tx] Xbee	Se utiliza para el envío de información a través del canal de transmisión	***Serial.begin*** **( )**
4	S	Sincronismo	Enciende un led indicando que los módulos se han sincronizado correctamente.	**"Led_Sinc"**
7	S	Izquierda activa	Enciende un led indicando que el intermitente izquierdo está encendido	**"Izquierda_ON"**
8	S	Derecha activa	Enciende un led indicando que el intermitente derecho está encendido.	**"Derecha_ON"**
12	S	Batería baja	Enciende un led indicando que la batería en el nodo está baja.	**"Led_Bateria"**

Tabla 27. Definición de variables globales Nodo Arduino.

5.3.2.4.1 Lectura de entradas.

En el Nodo, hemos definido dos tipos de entradas activas:

- Puerto Serie (R_x)
- Analógicas (conectadas a los sensores)

Dado que en el punto "*5.3.2.3 Puerto Serie: comunicación y sincronismo*" se ha indicado la operativa general para envío y recepción de datos por los pines 0 [**R_x**] y 1 [**T_x**] de la placa, en este punto nos centraremos en la **lectura de entradas analógicas**.

El controlador Arduino UNO dispone de 6 entradas analógicas (acceden a convertidores analógicos del hardware).

Cada una proporciona 10 bits de resolución, es decir **1024 valores** diferentes, y **por defecto miden 5V desde tierra**[36].

Los sensores que nos dan los valores de entrada están conectados en los pines {**A_0, A_1, A_2, A_3**}.

PROCESSING: Para leer el valor de las entradas

- Definimos un vector **contenidoanalog [ ]** que contiene el valor analógico [0 – 1024] de cada una de las entradas. Dado que la información que contiene en nuestro caso son números enteros, la definimos como "*int*".

```
int contenidoanalog [6 ] ;                    // Definimos un vector de longitud 6 números
```

- Utilizamos la instrucción "*analogRead()*"

```
for (i=0; i<6; i++) {
contenidoanalog [i ]= analogRead ( i );// Leo y grabo A0 en contenidoanalog[0] hasta A6.
}
```

PIN	TIPO	Nombre	Funcionalidad	Definición variable
A0	E	Joystick (derecha + izquierda)	Entrada con información del sensor Joystick (derecha e izquierda)	**"Lectura_A0"** (contenidoanalog[0])
A1	E	Emergencia	Entrada con información de pulsación señal de emergencia	**"Lectura_A1"** (contenidoanalog[1])
A2	E	Luces de freno	Entrada con información de pulsación de frenado	**"Lectura_A2"** (contenidoanalog[2])
A3	E	Batería	Entrada con información sobre el estado de carga de la batería	**"Lectura_A3"** (contenidoanalog[3])

Tabla 28. Lectura de entradas [actuadores] – Nodo Arduino

[36] Aunque por defecto mide 5V desde tierra, es posible cambiar el rango usando el pin ARF y código de programación.

Lectura A_0: Intermitentes.

El Joystick empleado dispone de una salida llamada URx conectada a A_0.

En reposo, el valor que proporciona a la entrada corresponde con el punto medio de la polaridad aplicada al Joystick.

Presionando hacia la derecha / izquierda, variamos el dato indicando el movimiento aplicado en el eje X (Derecha-Izquierda).

Hemos fijado un valor superior por encima del cual consideramos que queremos girar a la derecha. Así mismo hemos indicado un valor inferior por debajo del cual indicamos que queremos girar a la izquierda.

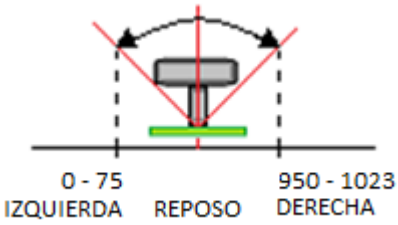
Acción	Valor A_0	Consecuencia
	[950 – 1023]	Variable Derecha = 1
	[76 – 949]	-
	[0 – 75]	Variable Izquierda = 1

Tabla 29. Lectura A_0 – Intermitentes 1

Con el comportamiento planteado, se define el flujograma de lectura de A_0.

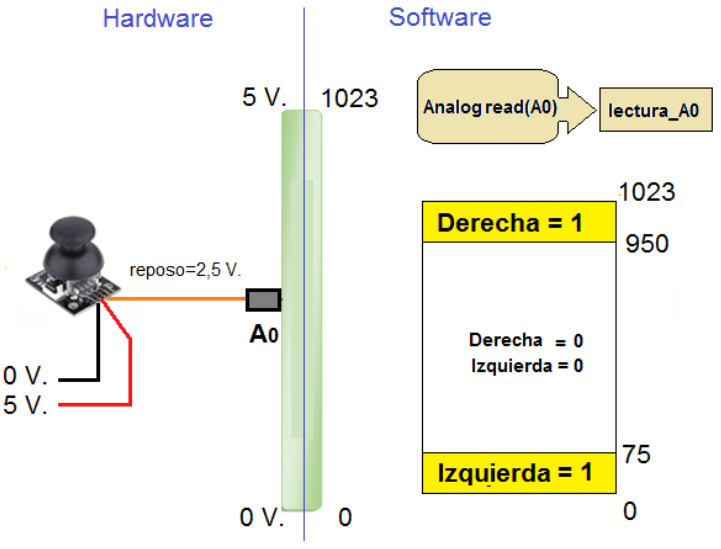
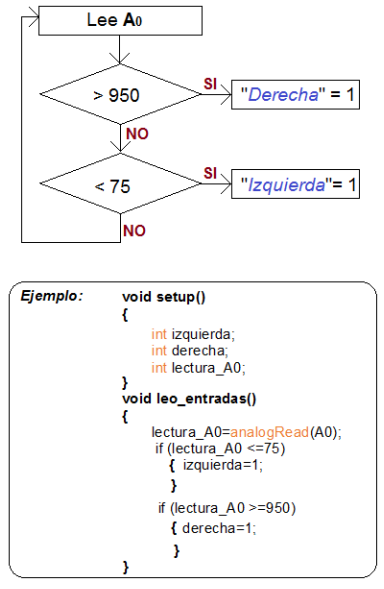
ENTRADA INTERMITENTES	***DIAGRAMA DE FLUJO***
	

Tabla 30. Lectura A_0 Intermitentes - 2

Lectura A_1: Emergencia.

La señal de emergencia viene de un interruptor que proporciona 5V en la entrada A_1 cuando está accionado. El valor de reposo es 0V.

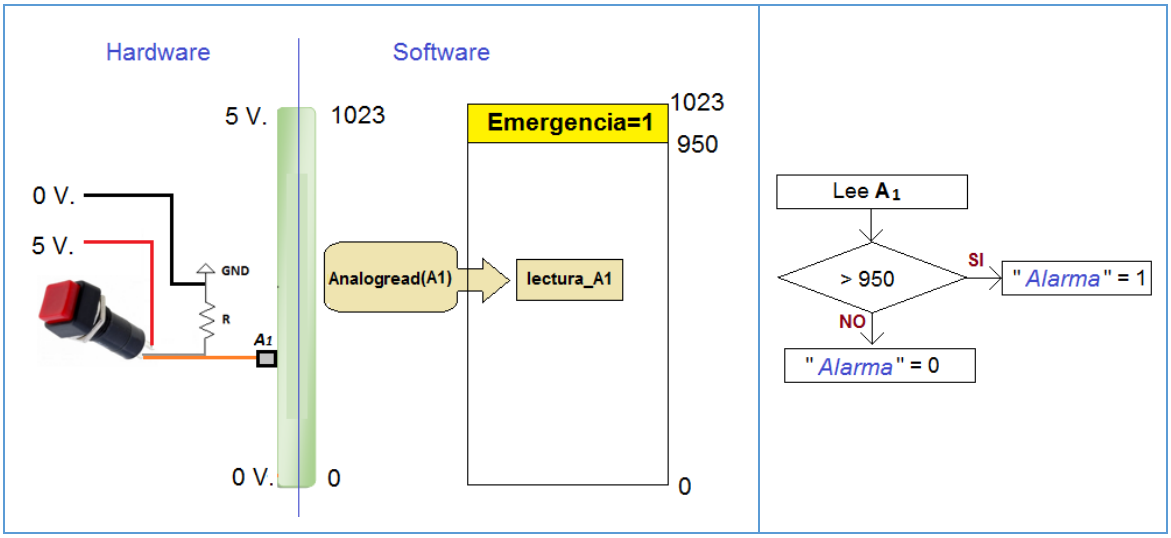
ENTRADA EMERGENCIA	DIAGRAMA DE FLUJO
	

Tabla 31. Lectura A_1 Emergencia

Lectura A_2: Freno.

La señal del freno se activa mediante un pulsador que está conectado en la entrada A2. Siempre que el pulsador esté actuado pondremos 0V y estaremos indicando que estamos frenando, como se indica en la figura

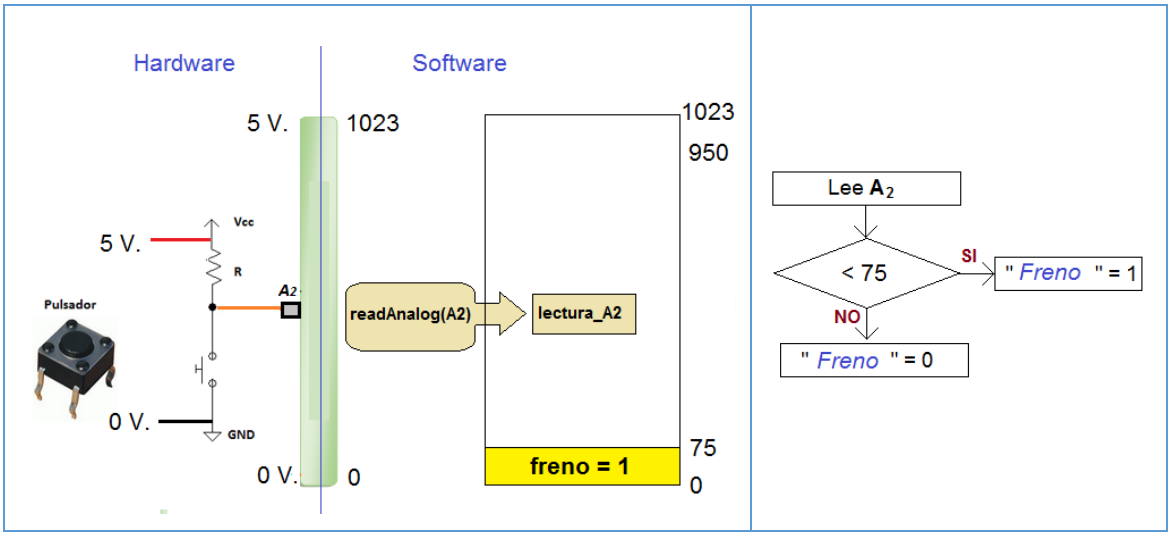
ENTRADA FRENO	DIAGRAMA DE FLUJO
	

Tabla 32. Lectura A_2 Freno

Lectura A_3: Batería.

Además del estado de los pulsadores, se incluye una señalización del estado de carga de la batería.

Para ello se ha incluido un divisor resistivo conectado directamente a la batería (Vcc=9V.) como se indica en la "*Tabla 33. Lectura A3* ".

Cuando está cargada (V_{cc} = 9V) la tensión en la entrada A3 del Arduino es de 5V. y cuando la consideramos descargada (V_{cc}=6V.), proporciona una tensión de 3,3V.

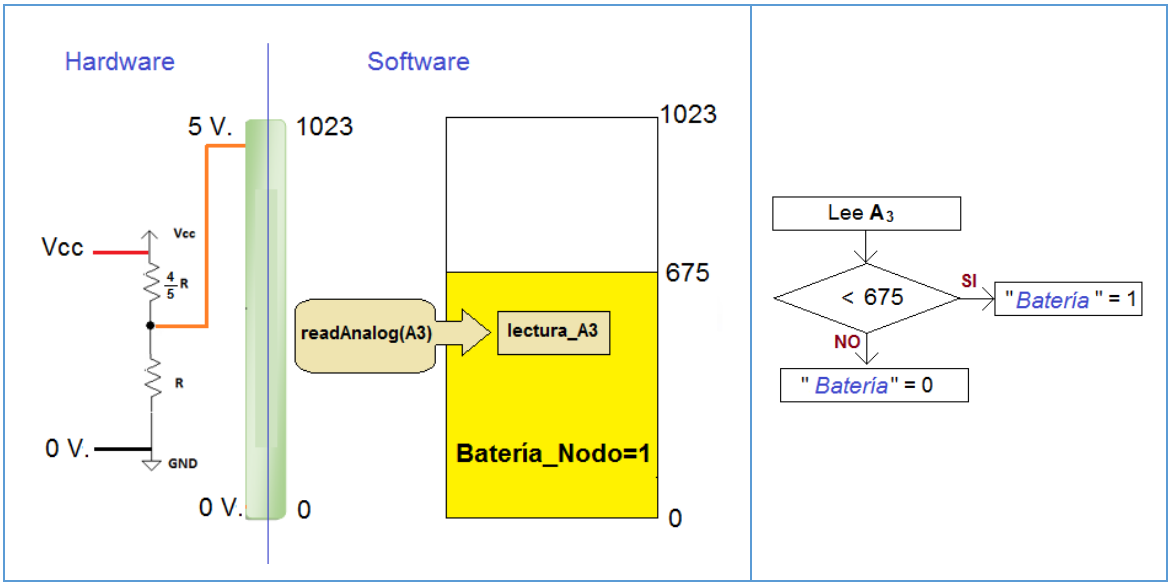
ENTRADA BATERÍA	**DIAGRAMA DE FLUJO**
	

Tabla 33. Lectura A_3 Batería

5.3.2.4.2 Escritura en salidas.

Al igual que en el caso de las entradas, podemos diferenciar:

- Escritura por el Puerto Serie (Tx)[37].
- **Digitales** (conectadas a leds de control): De los 14 pines que pueden usarse como E/S digital, utilizaremos los seis que no pueden configurarse para aportar funcionalidades específicas como comunicación por Puerto Serie, PWM, etc

  Los pines que configuramos como **salidas digitales** son **{4, 7, 8, 12}** y se utilizan para la monitorización de algunos estados del sistema mediante encendido de un led de control.

  Dichos estados son sincronismo, intermitentes y estado de la batería.

PROCESSING: Para escribir el valor de las salidas.

- Indicamos el comportamiento de los pines de salida mediante la función "pinMode":

  pinMode(4;OUTPUT);

- Definimos un vector "salidasdigitales [ ]" de dimensión 6 que contiene el valor de los pines definidos como salidas y una variable "contenidodig [ ]".

  int **salidasdigitales[]= {2, 4, 7, 8, 12, 13};**// Definimos un vector de longitud 6 números

  int **contenidodig[];**

- Para escribir el valor utilizamos la instrucción "*digitalWrite()*".

  digitalWrite(4;HIGH); // HIGH →Pongo un 1 ; LOW → Pongo un 0.

  digitalWrite(salidasdigitales(i);contenidodigital(i));

  // utilizo las variables definidas en el programa.

PIN	TIPO	Nombre	Funcionalidad	Definición variable
4	S	Sincronismo	Enciende un led indicando que los módulos se han sincronizado correctamente.	**"Led_Sinc"** (contenidodig[0])
7	S	Izquierda activa	Enciende un led indicando que el intermitente izquierdo está encendido	**"Izquierda_ON"** (contenidodig[1])
8	S	Derecha activa	Enciende un led indicando que el intermitente derecho está encendido.	**"Derecha_ON"** (contenidodig[2])
12	S	Batería baja	Enciende un led indicando que la batería en el nodo está baja.	**"Led_Bateria"** (contenidodig[3])

Tabla 34. Salidas digitales Nodo

[37] Tanto la recepción como la transmisión de datos [Rx, Tx] por puerto serie han sido incluidas en *"6.2.3 Puerto Serie: comunicación y sincronismo"*

Monitorización sincronismo

Con esta señal indicamos al ciclista que el sistema NODO-COORDINADOR está activo, es decir, que ambos módulos se comunican correctamente.

En el programa generamos una variable "***sinc***" que podrá tomar los valores 0 y 1.

Para determinar el estado, el nodo envía de forma cíclica una consulta al coordinador hasta recibir respuesta de sincronismo, momento en el que comienza a ejecutarse el resto del programa, de forma que:

- **"Sinc" = 1** → "*digitalWrite(4,HIGH)*".

  Si el coordinador envía la respuesta de ok sincronismo al nodo. Continúa la ejecución del programa y en el hardware del manillar se enciende un led indicando al ciclista que existe comunicación entre las partes delantera y trasera.

- **"Sinc" = 0** → "*digitalWrite(4,LOW)*".

  No se recibe respuesta del coordinador, por lo que no existe comunicación entre módulos. El led del hardware del manillar permanece apagado. La consulta de sincronismo continúa lanzándose cíclicamente hasta obtener respuesta.

En la tabla de la figura se muestra el diagrama de flujo de la actuación del led.

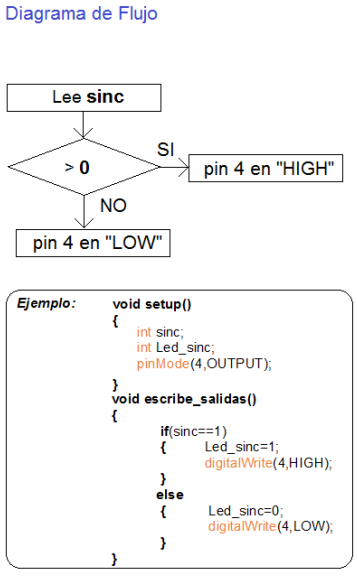
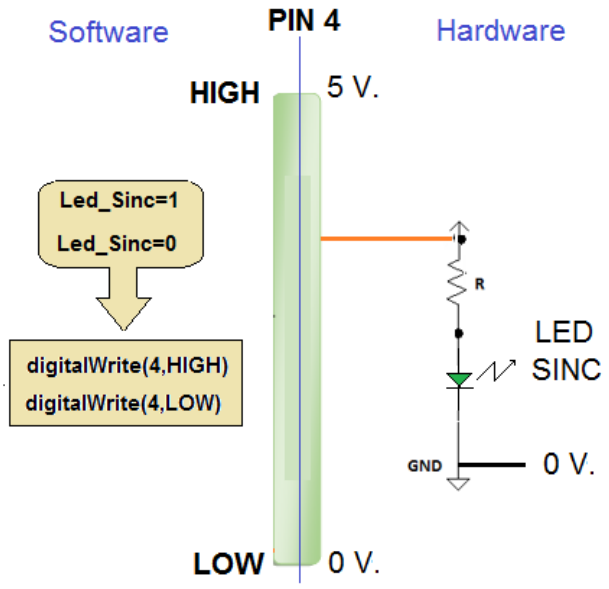
DIAGRAMA DE FLUJO	**SALIDA SINCRONISMO**
	

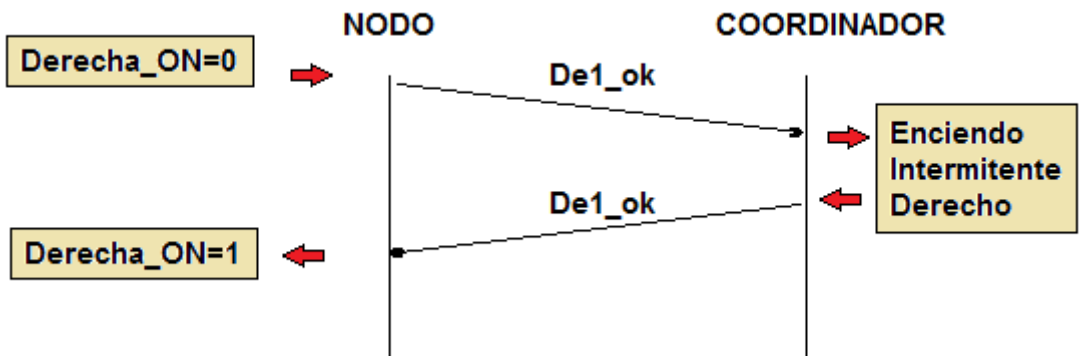
Tabla 35. Salida Pin4 – Sincronismo OK

Monitorización Intermitentes.

Para facilitar la interacción del usuario con el sistema, señalizamos en el hardware del manillar si alguno de los intermitentes está activo. Para ello habilitamos dos salidas conectadas a un diodo led:

- **pin 7: Izquierda.**
- **pin 8: Derecha.**

En la siguiente figura se muestra un ejemplo del proceso en el que el nodo envía la instrucción, el coordinador la ejecuta y envía un código de verificación.



Mediante las variables Derecha_ON e Izquierda_ON conocemos el estado de los intermitentes (encendido y apagado), y en consecuencia escribimos un 0 o un 1 como valor de las salidas asignadas.

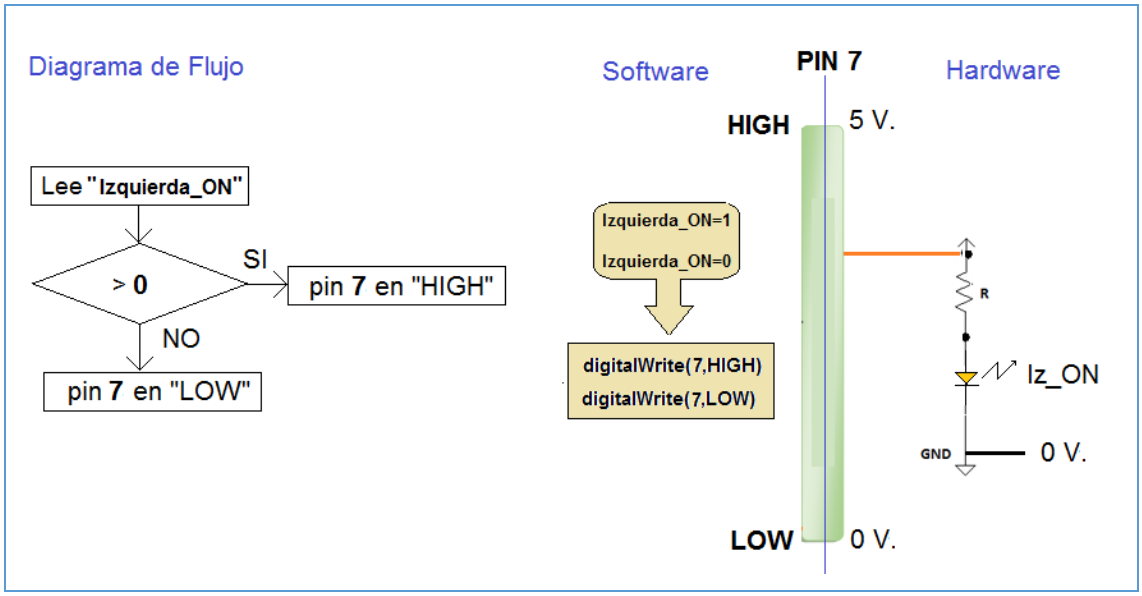


Tabla 36. Salida Pines 7 y 8 – Intermitentes.

Monitorización Batería

Para señalizar cuando la batería que alimenta el nodo está a un nivel bajo conectaremos un diodo led en el pin “12” de la placa del controlador.

El diagrama de flujo es el mostrado en la figura.

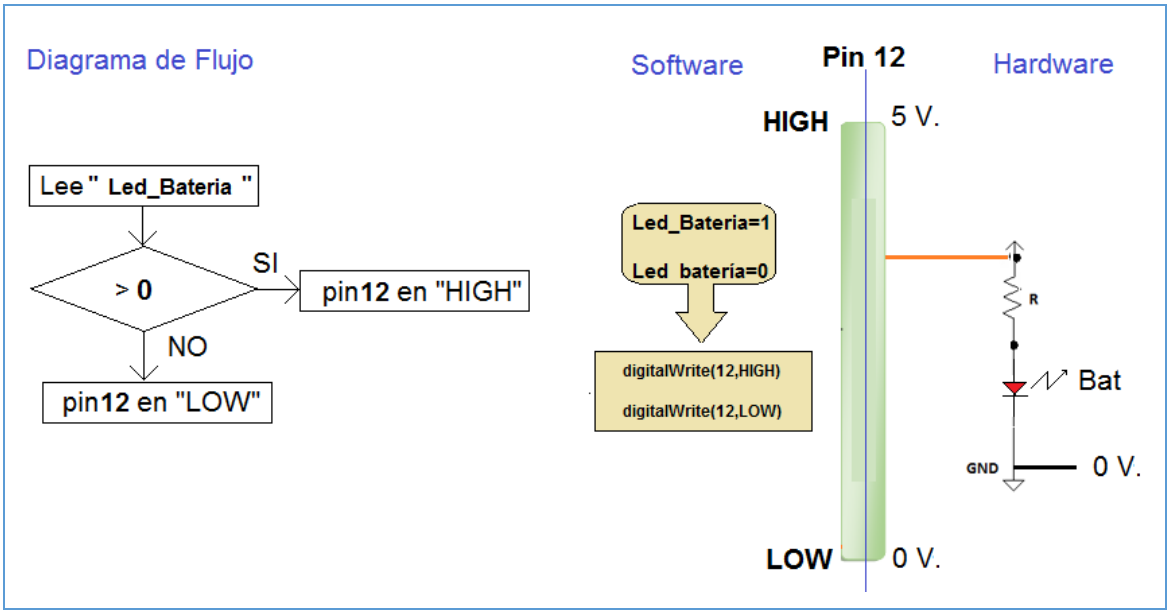


Tabla 37. Salida Pin 12 – Batería Nodo.

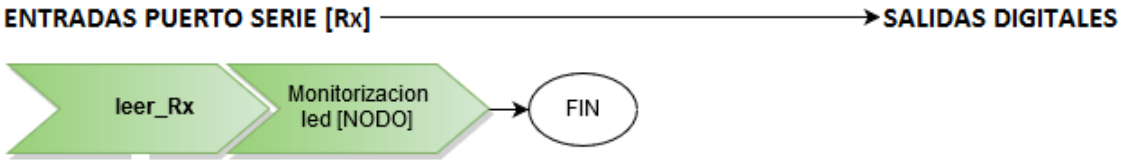
Se incluye como **propuesta de mejora futura** realizar una función que detecte el estado de la batería baja tanto del nodo como del coordinador, y lo comunique mediante el encendido único de un led.

5.3.2.4.3 Procesado información

Como procesado de información nos referimos a los programas y rutinas que realizamos para la modelización de los datos, desde la obtención de los mismos hasta su escritura.

Para ello, diferenciamos entre los siguientes casos:

1. Entradas Puerto Serie → Salidas Digitales
2. Entradas Analógicas → Salidas Puerto Serie



A través del puerto serie **[pin 0 – R_x]** llegan los datos procedentes del coordinador.

En nuestro caso particular nos informan sobre el estado del sincronismo y el de encendido de las luces intermitentes, de la siguiente manera:

Cod.						**Significado**	
/	**C01N01**	***I***	**z**	**X**	_ok	Intermitente Izquierdo	"*x*=1" encendido "*x*=0" apagado
/	**C01N01**	***D***	**e**	**X**	_ok	Intermitente Derecho	"*x*=1" encendido "*x*=0" apagado
/	**C01N01**	***S***	**i**	**X**	_ok	Solicitud de sincronismo	"*x*=1" sincronizado "*x*=0" no sincronizado

Tabla 38. Códigos de control izquierda, derecha y sincronismo.

Con los datos recibidos, si "X=1" encendemos el led de control correspondiente, en caso contrario, lo apagamos tal y como hemos estudiado en detalle anteriormente[38].

En la figura a continuación se muestra el diagrama de flujo correspondiente:

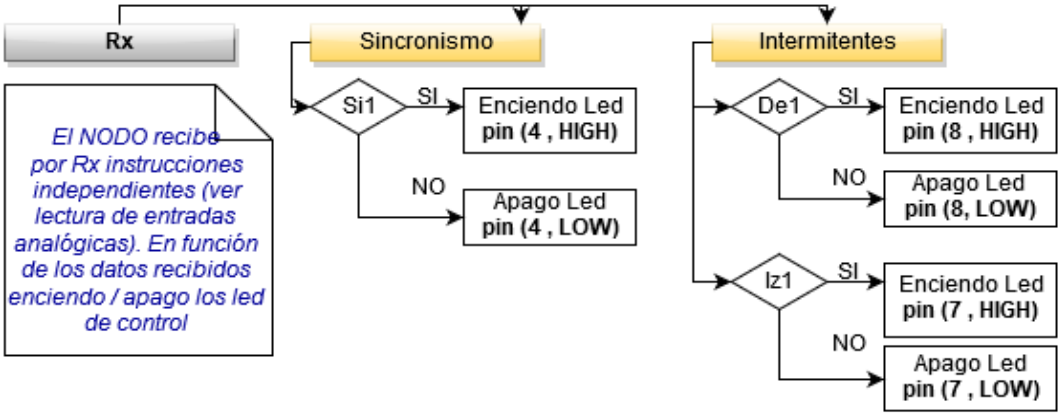


Ilustración 63. Nodo – Procesado entradas Rx

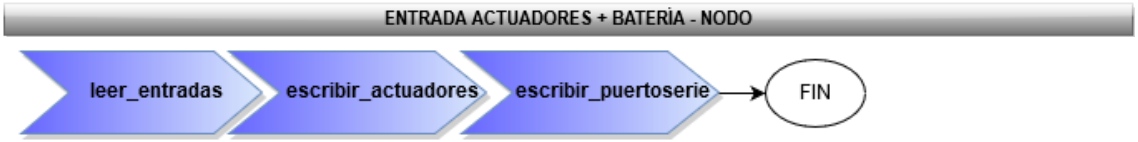
[38] Ver punto 6.2.4.2 "Escritura en salidas"

ENTRADAS ANALÓGICAS {A_0; A_1; ...} ⟶ **SALIDAS PUERTO SERIE [Tx]**

Para la programación del controlador ante cualquier cambio en las entradas correspondientes a los actuadores y a la batería, hemos diseñado tres rutinas:

- Rutina 1: "**leer_entradas**"
- Rutina 2: "**escribir_actuadores**"
- Rutina 3: "**escribir_puertoserie**"

De esta forma en el programa principal o **Loop ()** realizo las llamadas a las rutinas que correspondan.



leer_entradas

- *pines {A_0;…; A_3}:* entrada actuadores
- *"estado T (i)":* vector que contiene el valor de lectura actual
- *"estado T0 (i)":* vector que contiene el valor de lectura anterior
- *"cambio":* variable que contiene un número en función de si se ha producido cambio o no, y sobre qué actuación (freno, derecha, izquierda, batería o alarma)
- *Otros…* Variables globales: derecha, izquierda, emergencia, freno y bateria_nodo.

En "**leer_entradas**" se obtiene una lectura de los pines {A_0;…; A_3}.

La lectura se almacena en un vector "Estado T" además de actualizar las variables globales "*derecha*", "*izquierda*", "*emergencia*", "*freno*" y "*bateria_nodo*".

Dado que enviamos instrucciones al coordinador sólo cuando se detecta un cambio en nuestro programa (por ejemplo, pulso izquierda, pulso freno,…), realizamos una comparación entre "*Estado T*" y otro vector previamente definido "*Estado T_0*".

Si se ha producido alguna actuación, asignaremos a la variable "*cambio*" un valor que nos permita posteriormente (rutina *escribir_actuadores*) identificar la acción a realizar.

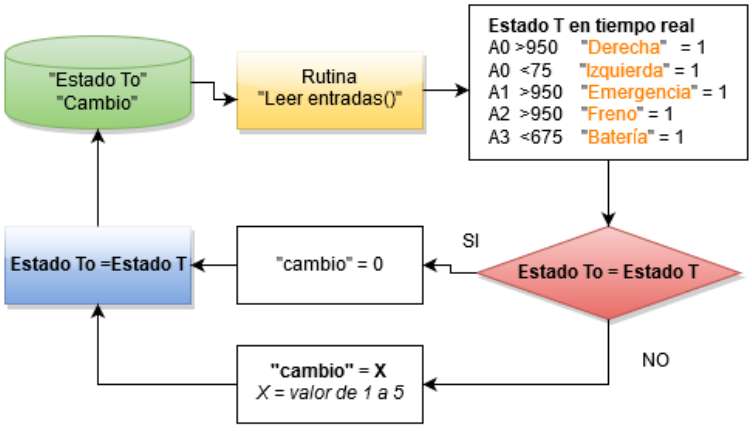


Ilustración 64. Nodo – Rutina "Leer entradas".

escribir_actuadores

- ❖ *"cambio":* variable que contiene un número en función de si se ha producido cambio o no, y sobre qué actuación (freno, derecha, izquierda, batería o alarma)
- ❖ *"codigo_orden (i)"* vector (3) que contiene el código de orden para enviar [T_x] al coordinador.
- ❖ *Otros…* Variables globales: derecha, izquierda, emergencia, freno y bateria_nodo.

En función del contenido de "*cambio*", se establece que:

- Si cambio = 5 → enciendo o apago el led de monitorización de batería.
- 0 < cambio < 5 → Envío al coordinador el código orden con la instrucción que corresponda.

"cambio"	***"codigo_orden"***			**Variable**		**Receptor**
5	"bateria_nodo"= 1 → pin(12;HIGH) "bateria_nodo"= 0 → pin(12;LOW)			*"bateria_nodo"*	*x*=1 encender *x*=0 apagar	**NODO**
4	*F*	r	X	"freno" = X	*x*=1 encender *x*=0 apagar	**COORDINADOR**
3	*E*	m	X	"emergencia" = X	*x*=1 encender *x*=0 apagar	**COORDINADOR**
2	*D*	e	X	"derecha" = X	*x*=1 encender *x*=0 apagar	**COORDINADOR**
1	*I*	z	X	"izquierda" = X	*x*=1 encender *x*=0 apagar	**COORDINADOR**
0	--	--	--	--	--	**COORDINADOR**

Con los requisitos anteriores se diseña el programa para que actúe según el flujograma:

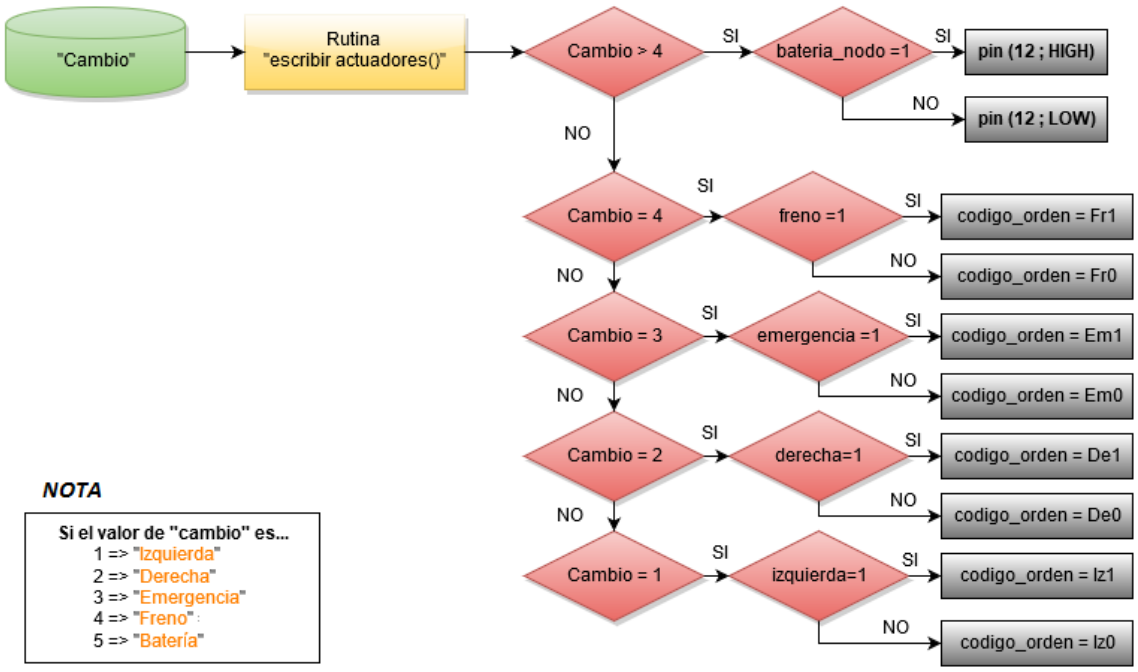


Ilustración 65. Nodo – Rutina "escribir_datos".

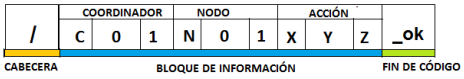
escribir_puertoserie

- *"codigo_orden(i)"* vector (3) que contiene el código de orden para enviar [T_x] al coordinador.
- *"codigo_ordenenviado(i)* vector (3) utilizado para verificación de datos.
- *"codigo_global(i)"* contiene "C01N01"
- *"codigo_fin(i)"* contiene "_ok"
- *"escribir":* variable cuyo contenido puede ser "0" o "1". Si es "0", inhabilita la escritura y termina la rutina. Si es "1" la escritura está habilitada y se puede proceder a escribir por Tx.

Esta rutina envía la instrucción completa al coordinador por el puerto serie (T_x).

La condición inicial[39] establece que la palabra completa debe contar con:

	Almacenado en…	TIPO	Contiene …
Cabecera	**--**	Fijo	"/"
Identificador	**"codigo_global"**	Fijo	"C01N01"
Acción	**"codigo_orden"**	Variable	Variable
Fin de código	**"codigo_fin"**	Fijo	"_ok"



Por lo tanto, una vez iniciada "*escribir_puertoserie*" si la variable de control "escribir" es distinta de 0 (no hay errores), procedemos enviar los caracteres por orden y a actualizar el valor de "*codigo_orden_enviado*":

Envío de datos:

- En primer lugar enviamos la cabecera ("/").
- Para enviar el resto de datos, utilizamos las variables "codigo_global", "codigo_orden" y "codigo_fin".

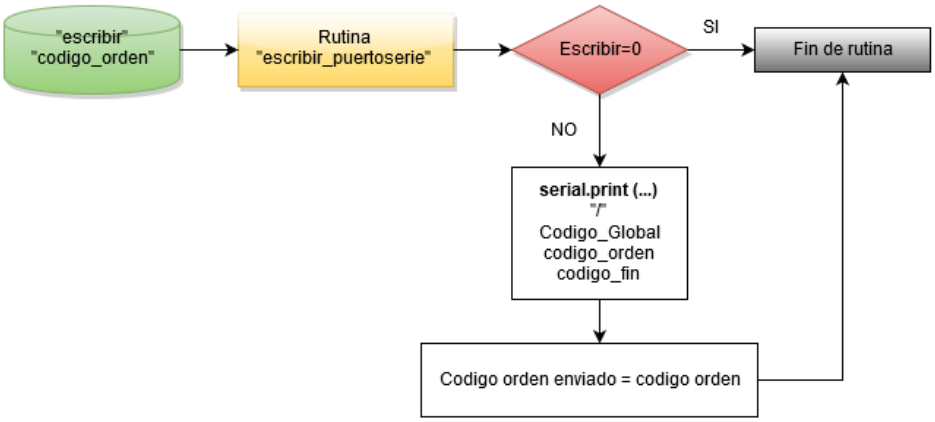


Ilustración 66. Nodo – Rutina "escribir_puertoserie"

[39] 5.2.2 Restricciones básicas.

5.3.2.5 MÓDULO COORDINADOR

Este módulo se ha diseñado para **recibir y procesar** los datos que proceden:

- De la batería.....{ **A_3**}
- Del nodo a través del radioenlace .....0 [**Rx**]

Por otro lado genera la señalización en los pines de **salida** para:

- Sincronizarse y enviar datos al Nodo .....1 [T**x**]
- Encender señales de intermitencia, alarma y freno .....{3, 5, 6}

PIN	TIPO	Nombre	Funcionalidad	Definición variable Setup ( )
A3	E	Batería	Entrada con información sobre el estado de carga de la batería	**"Lectura_CA3"**
0	E	[Rx] Xbee	Se utiliza para la recepción de información a través del canal de transmisión	-
1	S	[Tx] Xbee	Se utiliza para el envío de información a través del canal de transmisión	-
3	S	Señal Izquierda	B Enciende el conjunto de led's correspondientes a la señalización de giro a la izquierda.	**"PWM_Izquierda"**
5	S	Señal Derecha	Enciende el conjunto de led's correspondientes a la señalización de giro a la derecha.	**"PWM_Derecha"**
6	S	Señal Freno	Enciende el conjunto de led's correspondientes a la señalización de freno.	**"PWM_Freno"**

Tabla 39 Definición de variables globales Coordinador Arduino.

Según se muestra en la figura, mientras en el nodo el comportamiento general es que la información entra desde el hardware del sistema (pines analógicos) y las salidas se envían a través del puerto serie, en el coordinador ocurre al contrario: las entradas se reciben mayoritariamente por puerto serie y las salidas operan sobre los pines hardware.

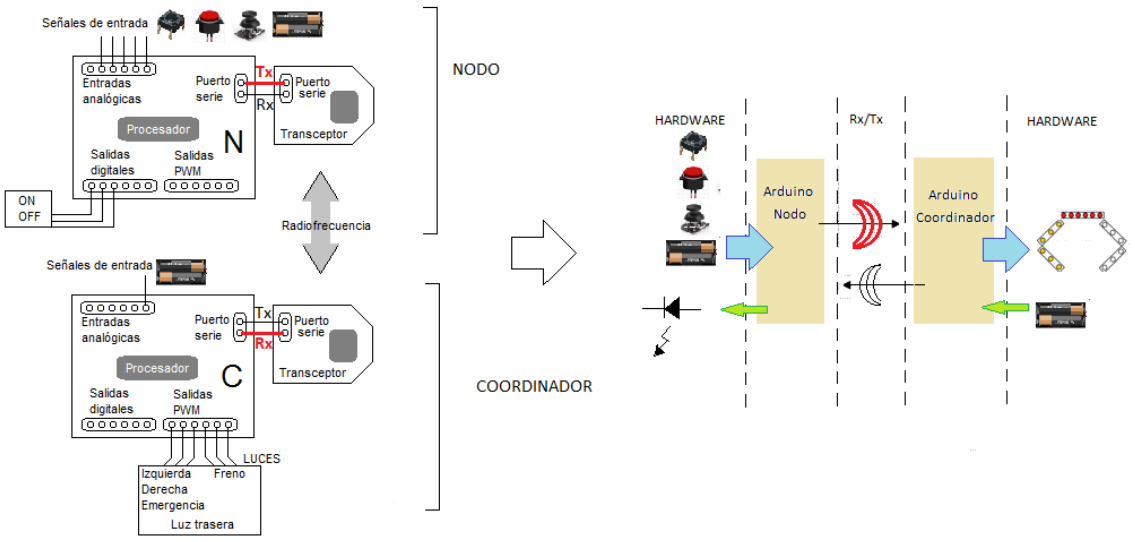


Ilustración 67. Comparación E/S en Nodo y Coordinador

5.3.2.5.1 Lectura de entradas

En el coordinador, hemos definido dos tipos de entradas:

- Analógicas (conectada a la batería)
- Puerto Serie (R_x)

PIN	TIPO	Nombre	Funcionalidad	Definición variable Setup ( )
A3	E	Batería	Entrada con información sobre el estado de carga de la batería	**"Lectura_CA3"**
0	E	[Rx] Xbee	Se utiliza para la recepción de información a través del canal de transmisión	**-**

Entradas analógicas:

Se ha reservado la entrada **A_3** para la mejora propuesta de monitorización del estado de la batería del coordinador, de forma simétrica al uso del pin en el Arduino - Nodo.

En el caso de implementarlo bastaría con incluir el mismo esquema que en el nodo, es decir, un divisor resistivo conectado directamente a la batería (ver apartado 5.3.2.4.1 Lectura de entradas.) y en la programación añadir:

- En el coordinador ..... una rutina que en el caso de que la batería baje de 6V envíe un código por puerto serie al nodo de "Batería baja"
- En el nodo..... una rutina que encienda el led de monitorización de batería si recibe la señal de batería baja del coordinador.

Entradas por puerto serie:

Como hemos indicado anteriormente[40] para la recepción de datos por el puerto serie es necesario definir los pines **[0]** y **[1]** de la placa Arduino como [**R_x**] y [**T_x**] mediante la instrucción "*Serial.begin* ()".

Para el tratamiento de los datos recibidos utilizaremos la rutina "**leer_puertoserie**"[41] en la que definimos un buffer para almacenar los datos recibidos y utilizamos las instrucciones "*serial.available()*" y "*serial.read()*" principalmente.

La rutina "leer_puertoserie", su correspondiente flujograma y el desglose de la información se incluye en el apartado "*5.3.2.5.3 Procesado* información"

[40] "*5.3.2.3 Puerto Serie: comunicación y sincronismo"*
[41] "*5.3.2.5.3 Procesado* información"

Procesado información".

5.3.2.5.2 Escritura de salidas

Las salidas en el coordinador pueden ser de dos tipos:

- Escritura por el Puerto Serie (Tx)[42].
- **PWM**: Si en las salidas digitales del nodo empleábamos aquellas que no podían configurarse con funcionalidades especiales, en este caso, utilizamos los pines que pueden configurarse para su uso como PWM.

PROCESSING: Ejemplo para pin PWM

- Definimos "voltaje_optimo" como el valor en el que la luminosidad del led vs consumo es óptima.

  int **voltaje_optimo**=225 // el valor 225 es un ejemplo aleatorio comprendido entre 0 y 255.

- Indicamos el comportamiento de los pines de salida mediante la función "pinMode":

  pinMode(3;OUTPUT);

- Indicamos que funciona como PWM:

  analogWrite(3;"voltaje_optimo")

PIN	TIPO	Nombre	Funcionalidad	Definición variable Setup ( )
1	S	[Tx] Xbee	Se utiliza para el envío de información a través del canal de transmisión	-
3	S	Señal Izquierda	B Enciende el conjunto de led's correspondientes a la señalización de giro a la izquierda.	**"PWM_Izquierda"**
5	S	Señal Derecha	Enciende el conjunto de led's correspondientes a la señalización de giro a la derecha.	**"PWM_Derecha"**
6	S	Señal Freno	Enciende el conjunto de led's correspondientes a la señalización de freno.	**"PWM_Freno"**

Tabla 40. Salidas Coordinador

[42] Tanto la recepción como la transmisión de datos [Rx, Tx] por puerto serie han sido incluidas en *"6.2.3 Puerto Serie: comunicación y sincronismo"*

Salidas PWM

Arduino dispone de 6 salidas con la numeración {3, 5, 6, 9, 10, 11} con capacidad de emitir señal en forma de un pulso modulado en amplitud (PWM) de **hasta 8 bits**, con una amplitud de "0V." a "5V." y un ancho del pulso comprendido entre los valores de "0" (0V.) y "255" (5V.).

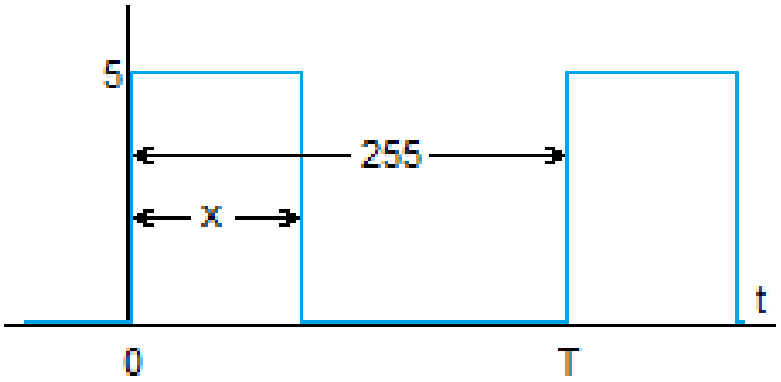


Ilustración 68. PWM

En nuestro proyecto utilizamos los **pines {3, 5, 6}** para el panel de indicaciones trasero: señalizar izquierda, derecha, alarma y freno.

El interés radica en que nos permite atenuar[43] los leds que conforman las señales **modulando el ancho de pulso**, y por lo tanto, podemos mejorar el rendimiento y el consumo del sistema estudiando con qué valor el resultado es más eficiente, es decir la sensación óptica es la misma utilizando la intensidad con ese ancho que con continua.

[43] No hemos implementado la funcionalidad de atenuación en nuestro programa, se plantea como mejora futura mediante la escritura de rutinas sin necesidad de hardware adicional.

5.3.2.5.3 Procesado información

Manteniendo la misma estructura para la presentación de las rutinas que en el caso del nodo, diferenciamos entre:

Para ello, diferenciamos entre los siguientes casos:

1. Entradas Analógicas → Salidas Puerto Serie
2. Entradas Puerto Serie → Salidas PWM

Respecto a las entradas analógicas, actualmente sólo se ha incluido la conexión a través del pin A3 de la batería.

Como se ha indicado anteriormente, se plantea como propuesta de mejora futura con objeto de realizar una mejora sobre la monitorización de la batería total.

No obstante, se incluye a continuación un esquema general del flujograma para su implementación.

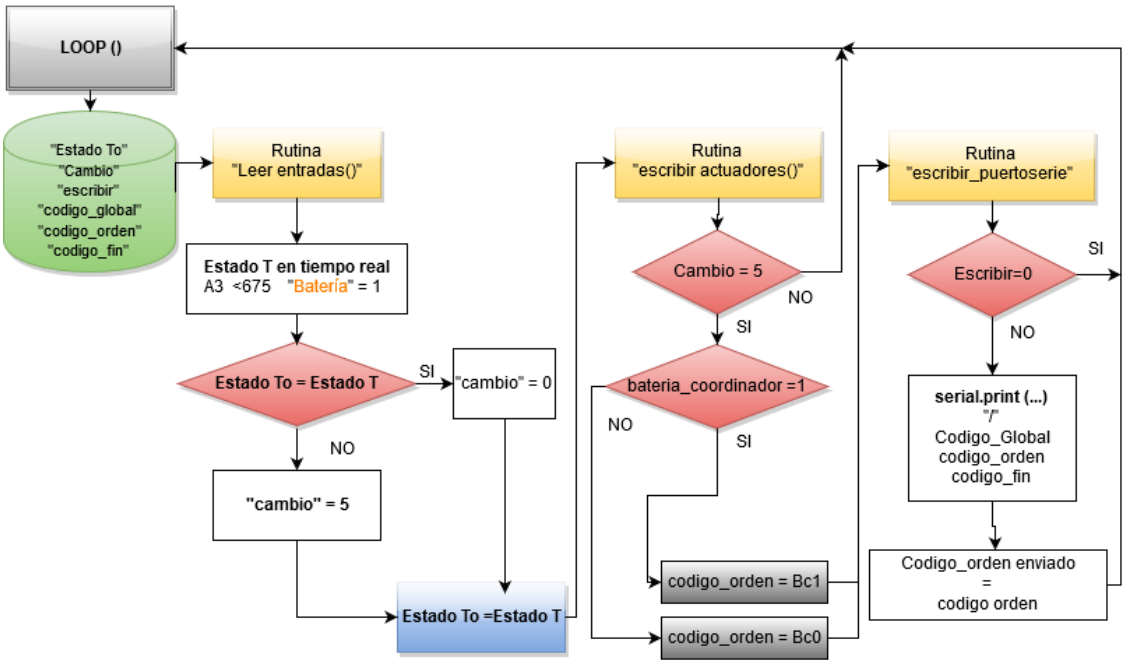
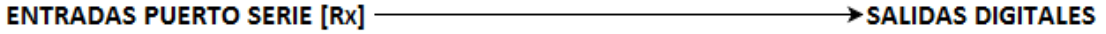


Ilustración 69. Coordinador - Monitorización batería



En este caso, hemos diseñado cuatro rutinas, de las cuales “ejecutar_orden” tiene otras 4 subrutinas anidadas tal y como se muestra en la figura a continuación:

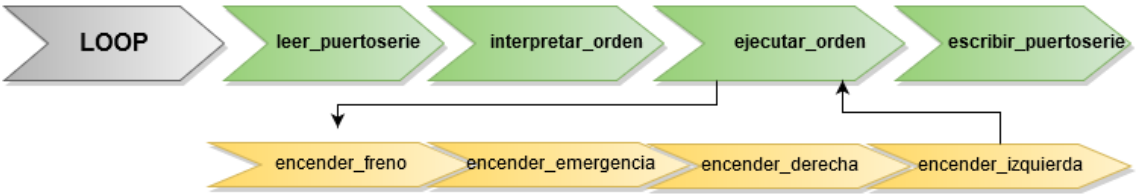


Ilustración 70. Coordinador – Entrada Rx [rutinas]

leer_puertoserie

- *dato[i]* entrada actuadores
- *“He_leido_datos”* variable con la confirmación de lectura de palabra ok (Error=0)
- *““codigo_orden(i)”* vector (3) que contiene el código de orden para enviar [T_x] al coordinador.
- *“codigo_global(i)”* contiene “C01N01”
- *“codigo_fin(i)”* contiene “_ok”

En “**leer_puertoserie**” se obtiene una lectura del bloque de información que previamente envió el nodo.

La lectura se almacena en un vector “dato[i]” que contiene la información necesaria para autenticar el bloque y para conocer la actuación que desea realizar el ciclista.

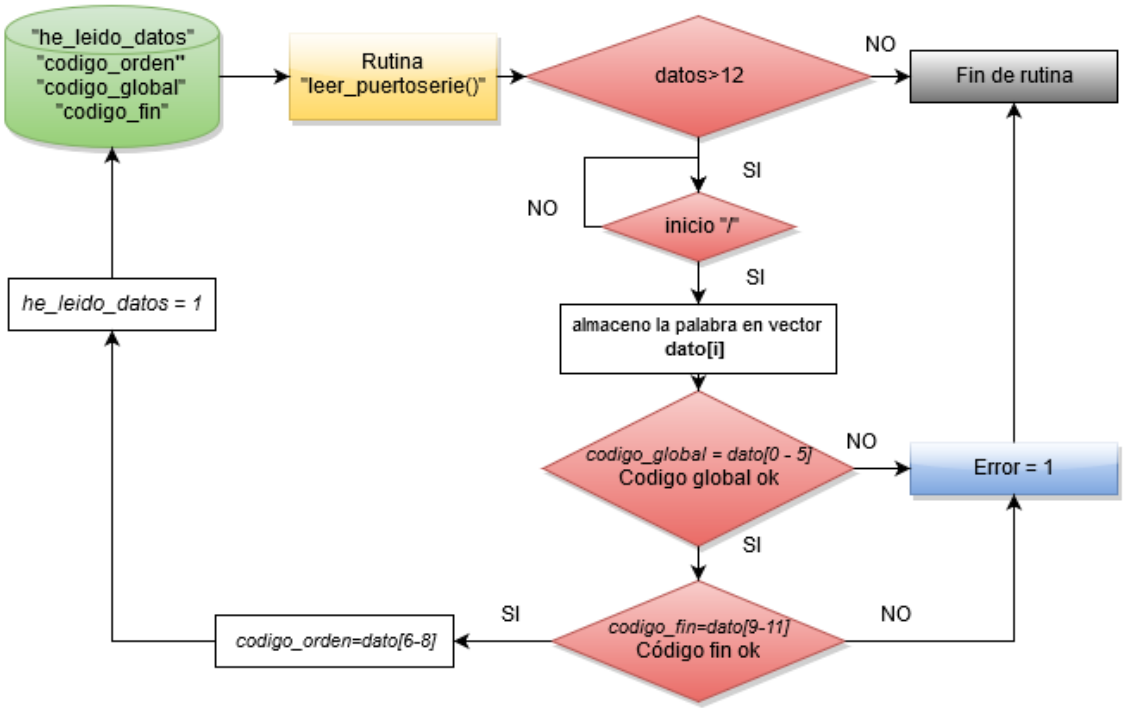


Ilustración 71 Coordinador – Rutina leer_puertoserie

interpretar_orden

- ❖ *"cambio":* variable que contiene un número en función de si se ha producido cambio o no, y sobre qué actuación (freno, derecha, izquierda, batería o alarma)
- ❖ *"código_orden (i)"* vector (3) que contiene el código de orden para enviar [T_x] al coordinador.
- ❖ *"estado T (i)":* vector que contiene el valor de lectura actual
- ❖ *"estado T0 (i)":* vector que contiene el valor de lectura anterior
- ❖ *Otros…* Variables globales: derecha, izquierda, emergencia, freno.

Recibimos instrucciones en el coordinador sólo cuando se detecta un cambio en el estado de las variables que implican actuación en el panel de señalización por lo que, realizaremos una comparación entre "*Estado T*" y otro vector previamente definido "*Estado T_0*".

El criterio seguido para fijar el contenido de "*cambio*" es el mismo que se estableció para el programa del nodo.

"cambio"	***"codigo_orden"***			**Variable**	
4	*F*	r	X	"freno" = X	*x*=1 encender *x*=0 apagar
3	*E*	m	X	"emergencia" = X	*x*=1 encender *x*=0 apagar
2	*D*	e	X	"derecha" = X	*x*=1 encender *x*=0 apagar
1	*I*	z	X	"izquierda" = X	*x*=1 encender *x*=0 apagar
0	--	--	--	--	--

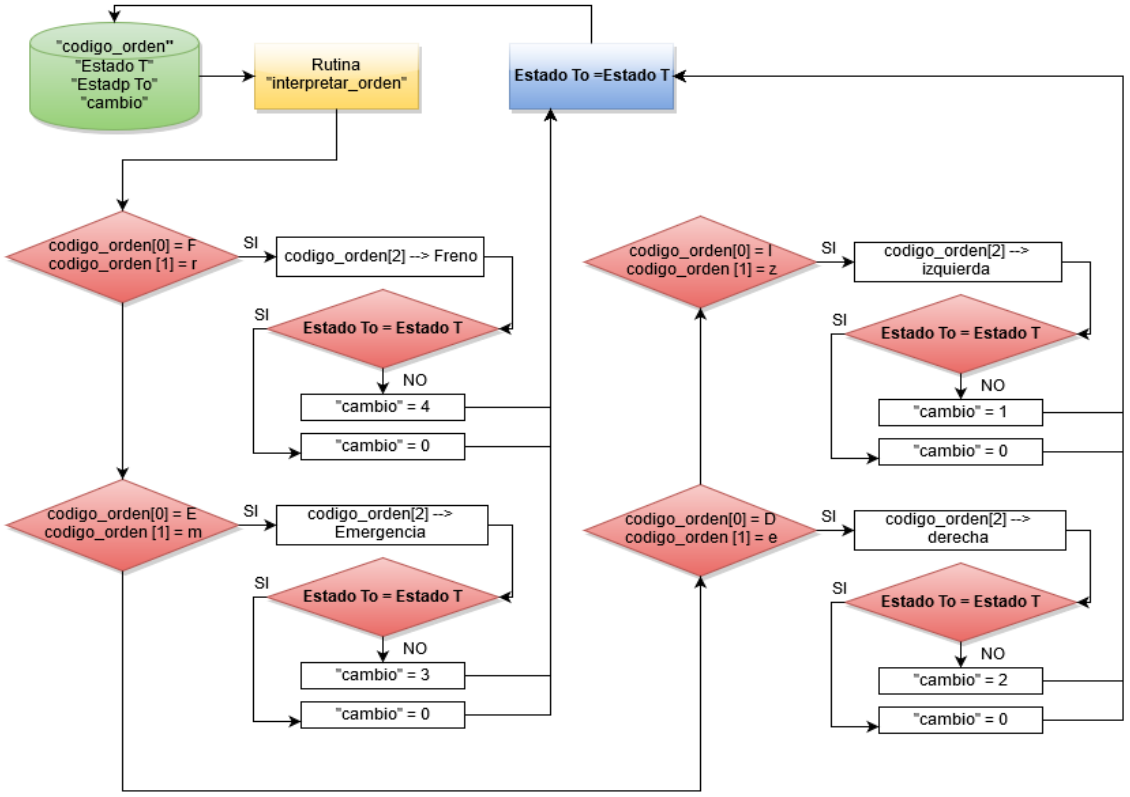


Ilustración 72. Coordinador – Rutina "interpretar_orden".

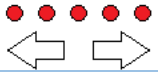
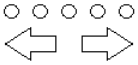
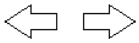
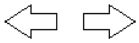
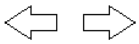
ejecutar_orden

- ❖ *"cambio":* variable que contiene un número en función de si se ha producido cambio o no, y sobre qué actuación (freno, derecha, izquierda, batería o alarma)
- ❖ *Otros…* Variables globales: derecha, izquierda, emergencia, freno.

Esta rutina se encarga de generar las órdenes que modifican el estado de los pines PWM del controlador Arduino para que se produzca el **encendido de la señal** correspondiente en el panel de señalización led del ciclista.

Para ello, disponemos de la variable "cambio" que nos aporta información sobre si existe variación o no y sobre qué elemento se ha producido.

Analizando conjuntamente "cambio" y las variables "freno", "emergencia", "derecha" e "izquierda" obtenemos la información necesaria para ejecutar las ordenes de encendido y apagado en los pines de la placa del coordinador:

Inicio...	Variables de entrada	Respuesta	Resultado final	
			Programa	Panel de señalización
	"Cambio"=4 "Freno"=X	*x*=1 encender	**analogWrite**(6,255)	
		x=0 apagar	**analogWrite**(6,0)"	
	"Cambio"=3 "emergencia" = X	*x*=1 encender	encender_emergencia();	
		x=0 apagar	analogWrite(5,0) analogWrite(3,0)	
	"derecha" = X	*x*=1 encender	encender_derecha();	
		x=0 apagar	analogWrite(5,0)	
	"izquierda" = X	*x*=1 encender	encender_izquierda();	
		x=0 apagar	analogWrite(3,0)	
0	--	--	--	--

Como las señales de alarma, izquierda y derecha implican un comportamiento de **intermitencia** (encendido y apagado de las luces de forma alternada) generamos unas sub-rutinas específicas que lo incluyen.

- "encender_emergencia()"
- "encender_derecha()"
- "encender_izquierda()"

Como se detalla a continuación, en la gestión de la intermitencia utilizamos unos temporizadores que marcan la frecuencia de encendido/apagado. Éstos temporizadores son iniciados en la rutina principal "ejecutar_orden" en el momento inmediatamente anterior a la llamada de las sub-rutinas.

Se diseña el programa para que actúe según el flujograma:

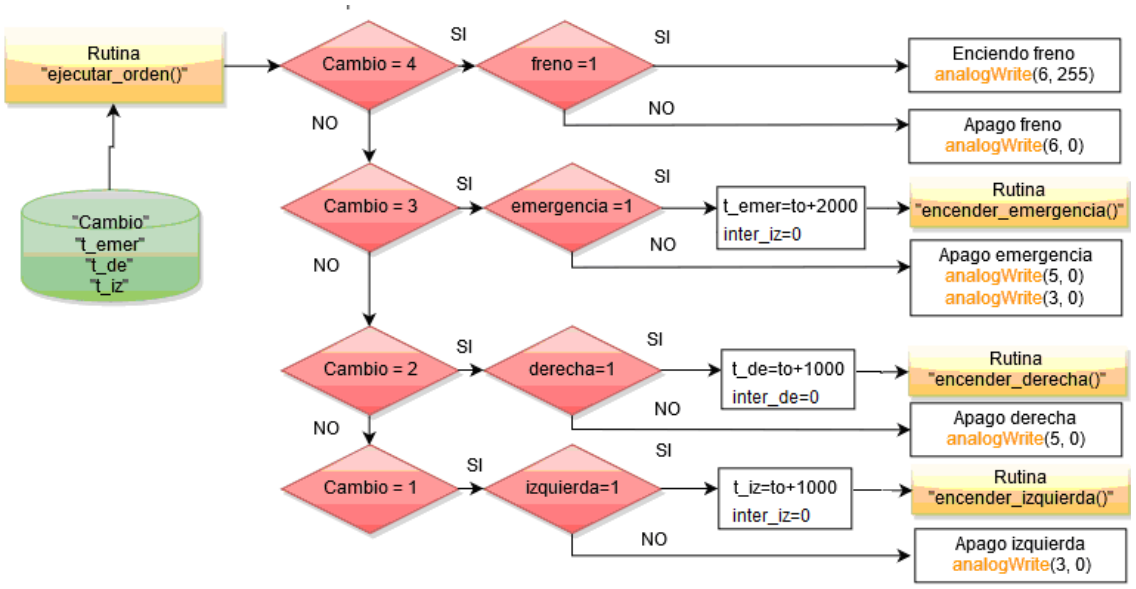


Ilustración 73. Coordinador - Rutina ejecutar orden

Efecto intermitencia.

El controlador Arduino tiene definida una rutina "***millis()***" que guarda el tiempo (en milisegundos) transcurrido desde el primer inicio del programa.

En nuestro diseño definimos una variable "**t_0**" cuyo valor se actualiza cada vez que comienza la rutina cíclica loop () ejecutando la instrucción

$$t_0 = millis();$$

Por otro lado, definimos:

Variable	Contenido	instrucción	Operación
"t_emer"	to + 2 segundos	**t_emer**=to+2000	Si t_emer>to+1000 → Encendido Si t_emer<to+1000 → Apagado
"t_de"	to + 1 segundo	**t_de**=to+2000	Si t_de>to+500 → Encendido Si t_de<to+500 → Apagado
"t_iz"	to + 1 segundo	**t_iz**=to+2000	Si t_iz>to+500 → Encendido Si t_iz>to+500 → Apagado

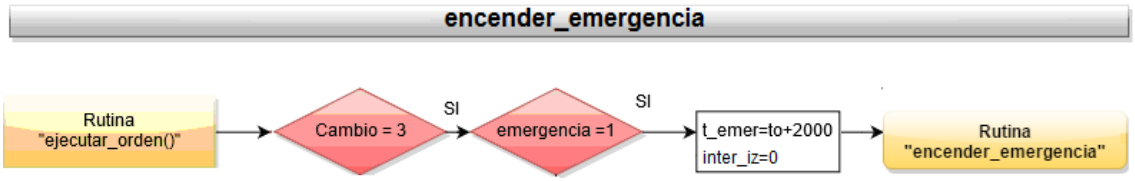
Cada vez que se da la orden de encender una señal que requiera de efecto de intermitencia, se asigna el valor de la columna "contenido" en la variable asociada.

Por ejemplo:

- Encender emergencia → t_emer=to+2000 → comparo t_emer con to+1000

De esta forma, si llega una instrucción de encender emergencia los leds se encenderán durante el tiempo que dure el ciclo de intermitencia.

Para la emergencia se ha definido que este intervalo sea de 2000 milisegundos (un segundo encendido y un segundo apagado), y para derecha e izquierda 1000 milisegundos (medio segundo encendido y medio segundo apagado).



En la rutina "ejecutar_orden()" si detectamos que se produce un cambio que implica encendido de la señal de emergencia ("cambio"=3 y "emergencia"=1), procedemos a guardar en la variable "t_emer" el contenido de t_0[44] más dos segundos.

t_emer=**to** +2000;

Una vez actualizado el dato, se ejecuta la rutina "**encender_emergencia**" cuyo flujograma es el que se muestra en la figura a continuación:

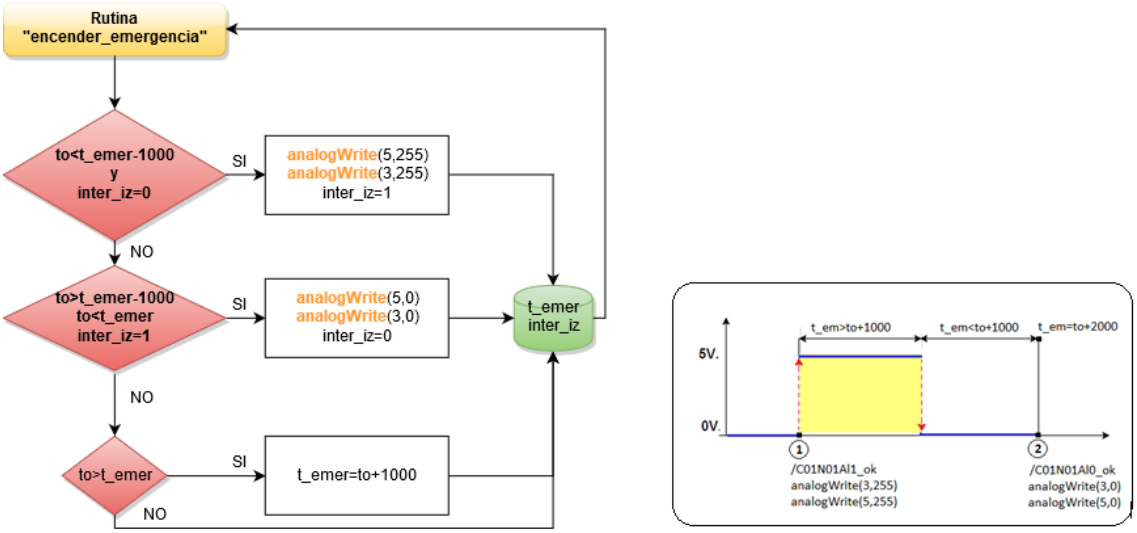


Ilustración 74. Coordinador – Rutina "encender_emergencia"

Para actualizar el valor de t_0 y escribir valores en pines 3 (izquierda) y 5 (derecha):

- En cada ciclo hasta que se apaga la señal de emergencia, se obliga a que el programa pase por esta rutina y se analiza el valor del tiempo "t_0".
    - Si **t_0**<t_emer-1000, nos encontramos en la zona marcada en amarillo en nuestra gráfica que corresponde con el encendido de la luz de emergencia (izquierda y derecha simultáneamente)
    - Si t_emer-1000<**t_0**<t_emer, corresponde a la zona de apagado (derecha e izquierda apagados).
    - Si t_0>t_emer, actualizamos el valor de t_emer (t_emer=t_0+2000) con lo que se comienza de nuevo al ejecutar en el siguiente ciclo la rutina..

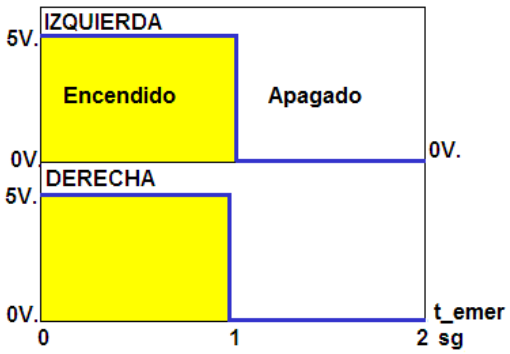
[44] T_0 es el tiempo de ejecución del ciclo del programa.

Variable “**inter_iz**”

Sólo se envía la orden de encendido (analogWrite(5;255) y analogWrite(3;255)) una vez, en el cambio.

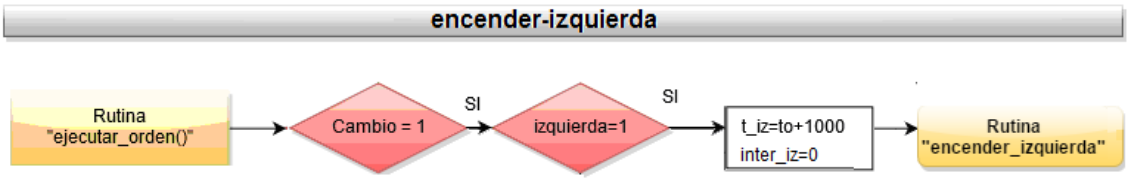
Para esto se usa una variable (inter_iz). Modificando su valor en 1 o 0 en función del estado de la intermitencia, regulamos que sólo se ejecute una vez cada escritura en el hardware.

En el dibujo de la figura, se muestra un diagrama temporal de encendido de las luces de emergencia y la rutina que lo ejecuta.




```
void encender_emergencia()
{
  if((to < (t_emer - 1000)) && inter_iz==0)
  {
      analogWrite(salidapwm[0],255);
      analogWrite(salidapwm[1],255);
      Serial.println("");
      Serial.print("emergencia on");
      inter_iz=1;
  }
  if( (to > (t_emer - 1000)) && (to < t_emer) && inter_iz==1)
  {
      analogWrite(salidapwm[1],0);
      analogWrite(salidapwm[0],0);
      Serial.println("");
      Serial.print("emergencia off");
      inter_iz=0;
  }
  if(to>t_emer)
  {
    t_emer=(to+2000);
    inter_iz=0;
  }
}
```

Ilustración 75. Funcionamiento luces de emergencia.



Cuando se activa el intermitente izquierdo en la rutina "ejecutar_orden()" ("cambio"=1 y "izquierda"=1) se guarda en la variable de tiempo "t_iz" el tiempo de ejecución del ciclo del programa más un segundo

t_iz=to +1000;

Una vez actualizado el dato, se ejecuta la rutina "**encender_izquierda**". En el flujograma de la figura se muestra el comportamiento:

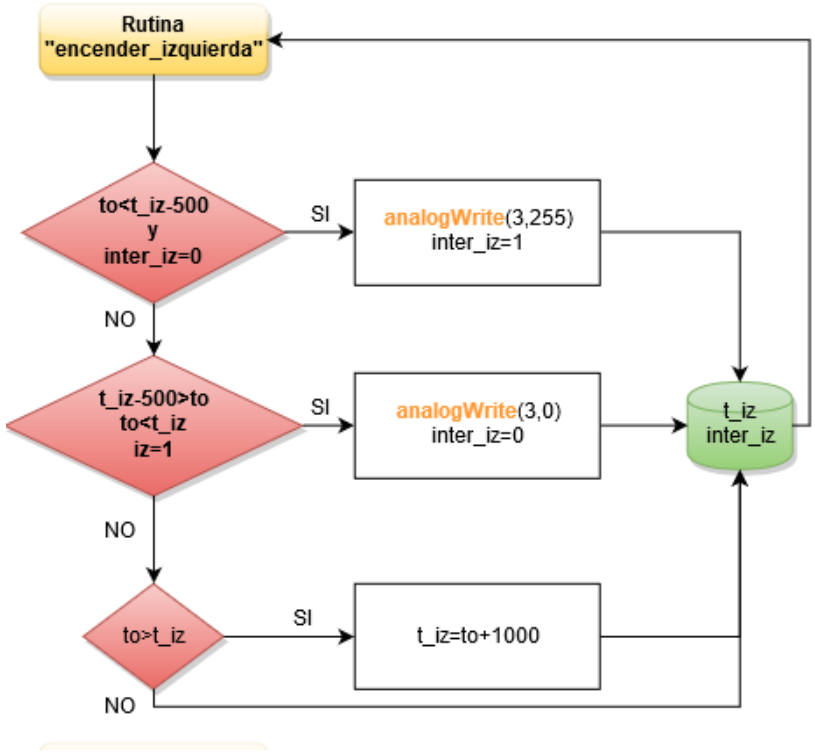


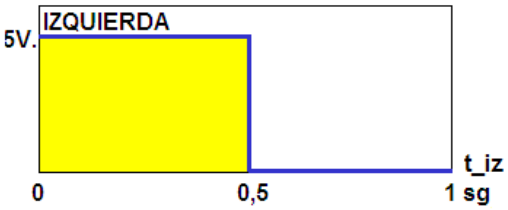
Ilustración 76. Coordinador – Rutina "encender_izquierda"

Para actualizar el valor de t_0 y escribir valores en pin 3 (izquierda):

- En cada ciclo hasta que se apaga la señal de emergencia, se obliga a que el programa pase por esta rutina y se analiza el valor del tiempo "t_0".
  - Si se encuentra en el intervalo **to<t_iz-500**, se encienden las luces de izquierda y se apagan las luces de la derecha si estuvieran encendidas.
  - Si se encuentra en el intervalo **t_iz-500<to<t_iz**, se apagan ambas luces (derecha y la izquierda).
  - Si se encuentra en **to>t_iz**, se actualiza t_iz con el valor del tiempo de ejecución del ciclo más un segundo (to+1000), con lo que se comienza de nuevo al ejecutar en el siguiente ciclo la rutina.

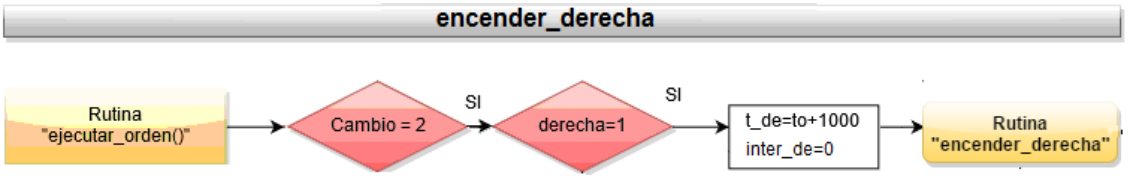
Sólo se envía la orden de encendido analogWrite(3;255)) una vez, en el cambio.

En el dibujo de la figura, se muestra un diagrama temporal de encendido de intermitente izquierdo y la rutina que lo ejecuta.




```
void encender_izquierda()
{
  if((to < (t_iz - 500)) && inter_iz==0)
  {
      Serial.println("");
      Serial.print("izquierda on");
      analogWrite(salidapwm[0],255);
      inter_iz=1;
  }
  if( (to > (t_iz - 500)) && (to < t_iz) && inter_iz==1)
  {
      Serial.println("");
      Serial.print("izquierda off");
      analogWrite(salidapwm[0],0);
      inter_iz=0;
  }
  if(to>t_iz)
  {
    t_iz=(to+1000);
  }
}
```

Ilustración 77. Funcionamiento intermitente izquierdo.



Cuando se activa el intermitente derecho en la rutina “ejecutar_orden()” (“cambio”=1 y “derecha”=1) se guarda en la variable de tiempo “t_de” el tiempo de ejecución del ciclo del programa más un segundo

t_de=to +1000;

En el flujograma de la figura se muestra el comportamiento

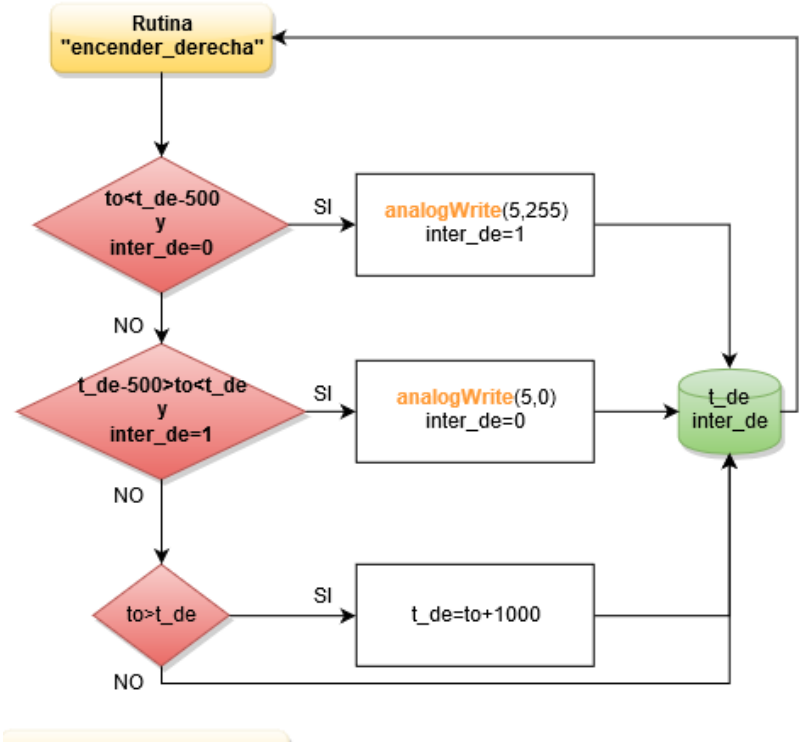


Ilustración 78. Coordinador - Rutina encender_derecha

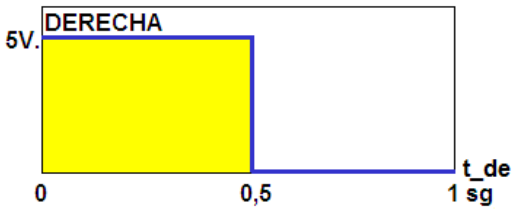
Para actualizar el valor de t_0 y escribir valores en pin 5 (derecha):

- En cada ciclo hasta que se apaga la señal de emergencia, se obliga a que el programa pase por esta rutina y se analiza el valor del tiempo “t_0”.
    - Si se encuentra en el intervalo to<t_de-500, se encienden las luces del intermitente derecho y se apagarán las luces del intermitente izquierdo.
    - Si se encuentra en el intervalo t_de-500<to<t_de, se apagan las luces del intermitente derecho.

- Si se encuentra en to>t_de, se actualiza t_de con el valor del tiempo de ejecución del ciclo más un segundo (to+1000), con lo que se comienza de nuevo al ejecutar en el siguiente ciclo la rutina.

Sólo se ejecuta la acción de encendido una vez, en el cambio. Para esto se usa una variable (inter_de) que regula que sólo se ejecute una vez cada escritura en en el hardware.

En el dibujo de la figura, se muestra un diagrama temporal de encendido de intermitente izquierdo y la rutina que lo ejecuta.




```
void encender_derecha()
{
  if((to < (t_de - 500)) && inter_de==0)
  {
      Serial.println("");
      Serial.print("derecha on");
      analogWrite(salidapwm[1],255);
      inter_de=1;
  }
  if( (to > (t_de - 500)) && (to < t_de) && inter_de==1)
  {
      Serial.println("");
      Serial.print("derecha off");
      analogWrite(salidapwm[1],0);
      inter_de=0;
  }
  if(to>t_de)
  {
    t_de=(to+1000);
  }
}
```

Ilustración 79. Funcionamiento intermitente derecho

escribir_puertoserie

- *"codigo_orden(i)"* vector (3) que contiene el código de orden para enviar [T_x] al coordinador.
- *"codigo_ordenenviado(i)* vector (3) utilizado para verificación de datos.
- *"codigo_global(i)"* contiene "C01N01"
- *"codigo_fin(i)"* contiene "_ok"
- *"escribir":* variable cuyo contenido puede ser "0" o "1". Si es "0", inhabilita la escritura y termina la rutina. Si es "1" la escritura está habilitada y se puede proceder a escribir por Tx.

Esta rutina envía información al nodo con la confirmación de que ha ejecutado órdenes, se ha sincronizado o su batería es baja.

Para ello, usamos el pin 1 de la placa de Arduino, correspondiente al puerto serie (T_x).

Las condiciones establecidas para el envío e interpretación de las palabras son idénticas a las desarrolladas en la rutina "escribir_actuadores" apartado "5.3.2.4.3 Procesado información" para el Nodo.

Por lo tanto, una vez iniciada "*escribir_puertoserie*" si la variable de control "escribir" es distinta de 0 (no hay errores), procedemos enviar los caracteres por orden y a actualizar el valor de "*codigo_orden_enviado*":

Envío de datos:

- En primer lugar enviamos la cabecera ("/").
- Para enviar el resto de datos, utilizamos las variables "codigo_global", "codigo_orden" y "codigo_fin".

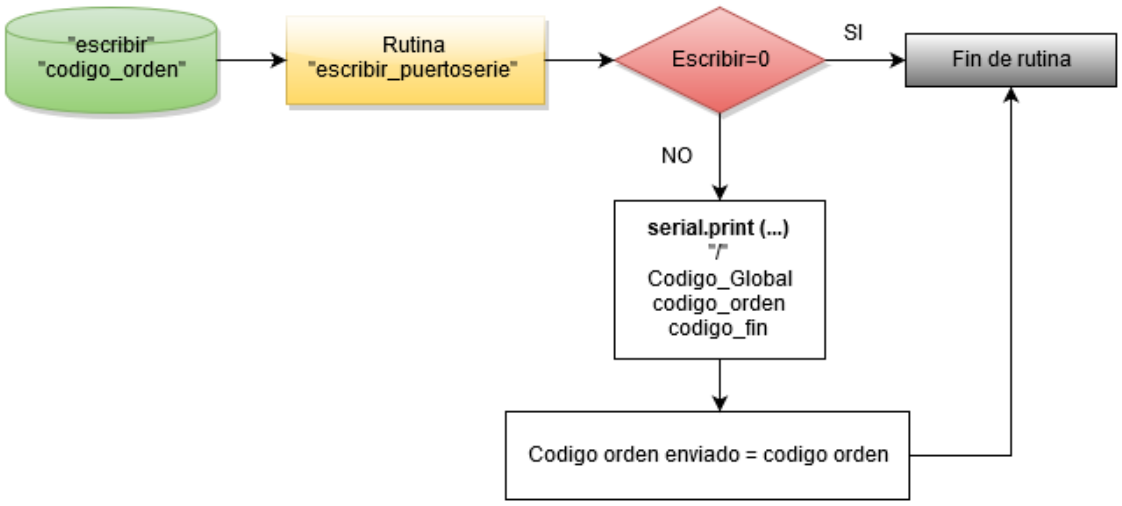


Ilustración 80. Coordinador – Rutina "escribir_puertoserie"

6 PRUEBAS Y VALIDACIÓN

La realización de este proyecto incluye la implementación de un piloto con las funcionalidades descritas anteriormente.

En este apartado se recoge el protocolo con las pruebas realizadas a los distintos módulos tanto de forma individual para comprobar su funcionalidad como de forma global una vez integrados en el sistema.

6.1 TRANSMISIÓN/RECEPCIÓN

Los transceptores conforman el Canal radio por el que se intercambia información.

Para probar el funcionamiento y configuración de los módulos Xbee se monta un sistema según se muestra en la figura:

Elementos Hardware	Unidades	Utilidad
Módulos XBee	2	**Transmisión**
Módulo Arduino UNO	1	**Controlador**
Ordenador portátil.	1	**Programación de módulos** • X – CTU para Xbee • Arduino IDE para el controlador.

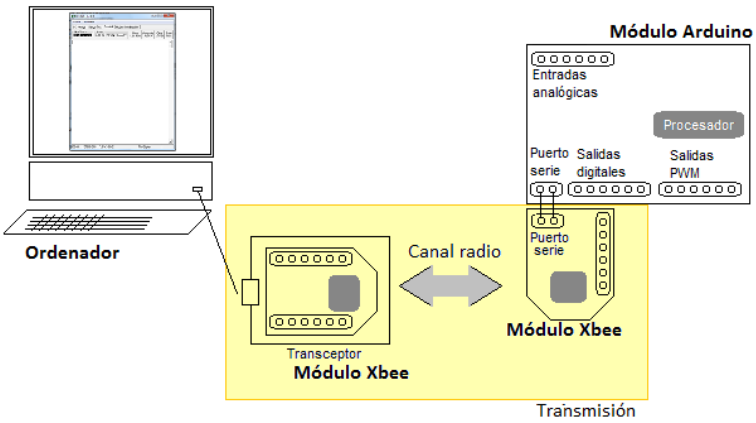


Ilustración 81. Banco de pruebas Xbee

CONFIGURACIÓN DEL CONTROLADOR – PRUEBAS XBEE.

Para realizar las pruebas se ha desarrollado un programa sencillo en el que el controlador contesta cada vez que recibe un carácter cualquiera por el puerto serie.

1. Ordenador envía mensaje a Arduino.
2. Arduino contesta al mensaje y se muestra en pantalla en el ordenador.

Si tanto la emisión como la recepción se producen según el diseño, nuestros módulos Xbee están funcionando correctamente.

CONFIGURACIÓN MÓDULOS XBEE.

Para la configuración y monitorización de los módulos XBee utilizamos desde el PC en el programa X-CTU cuyo interfaz es el que se muestra en la "Ilustración 82. Programa (Arduino) para pruebas Xbee".

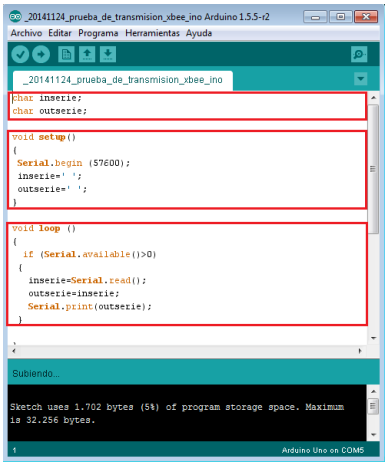


Ilustración 82. Programa (Arduino) para pruebas Xbee

Los módulos deben funcionar como un cable físico que transporta la información entre el puerto serie de la placa Arduino UNO y el Ordenador en las pruebas individuales, y entre 2 placas Arduino UNO (controladores) en el sistema global.

Por lo tanto, se escoge una configuración tipo "Conexión Transparente", cuyos principales parámetros de configuración son:

Comando	Valor Coordinador	Valor Nodo
PAN ID	3332	3332
MY	612	613
DL	613	612
RR	0	0
CE	1	0
A1	0	0
A2	0	0
BD	6	6
RO	0	0

Tabla 41. Parámetros configurados XBee durante las pruebas de funcionamiento.

RESULTADO.

- El ordenador envía un carácter al controlador Arduino a través del Xbee.
- El controlador Arduino recibe el carácter y contesta enviando el mismo carácter a través del XBee.

Utilizamos el programa XCTU para monitorizar el flujo de información.

Funciona correctamente cuando observamos en el puerto serie del XBee tanto el carácter enviado como el recibido.

- **Transmisión**: por Tx sale un carácter (**en azul**).
- **Recepción**: por Rx se recibe un carácter (**en rojo**).

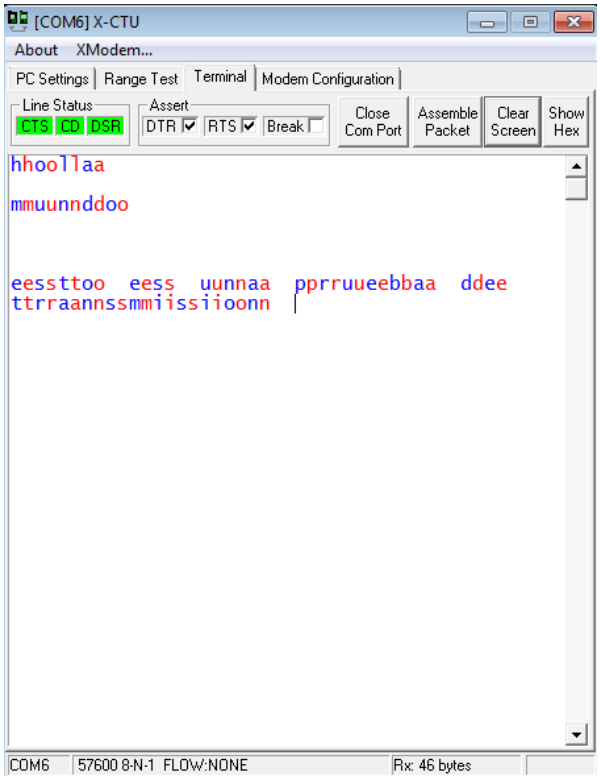


Ilustración 83. Pruebas – Monitorización Rx-Tx XBee

6.2 MODULO DE CONTROL

El módulo de control consta de dos placas Arduino UNO a las que denominamos controladores NODO y COORDINADOR.

Para su funcionamiento se ha desarrollado un programa en función de las entradas/salidas conectadas y el comportamiento deseado.

- El controlador **NODO**, situado en la bicicleta (manillar), capta las señales de las entradas (actuadores) y se encarga de procesar la información y enviar las órdenes correspondientes al controlador COORDINADOR, situado en el ciclista.

- El **COORDINADOR**, en función de las órdenes recibidas del NODO, procesa la información recibida y activa la señalización led correspondiente. Posteriormente envía confirmación al NODO de que todo el proceso se ha realizado correctamente.

CONTROLADOR NODO

El banco de pruebas implementado para probar el funcionamiento y configuración del controlador NODO es el que se muestra en la figura:

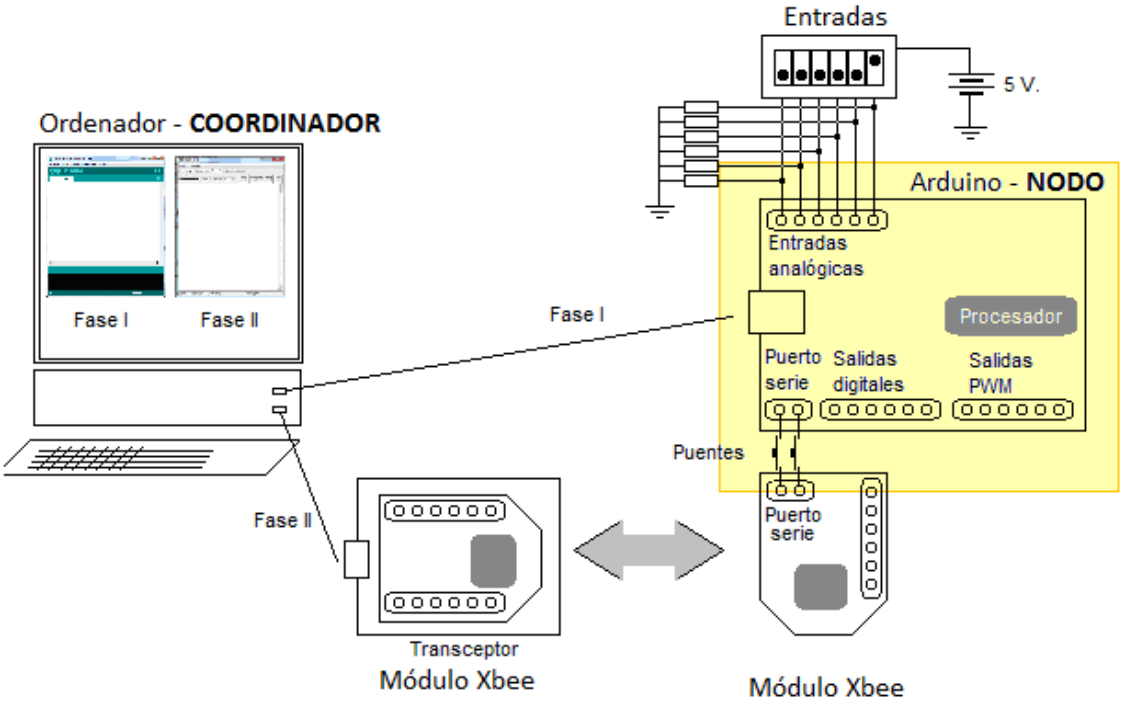


Ilustración 84. Banco de pruebas – controlador NODO

Los elementos auxiliares utilizados para las pruebas del Arduino NODO según se ha indicado en la Ilustración 84. Banco de pruebas – controlador NODO son:

- Entradas: ..... Para las entradas podemos bien utilizar los pulsadores y el Joystick, bien simular su comportamiento poniendo los valores de tensión correspondientes en los pines correspondientes a las entradas analógicas en la placa de Arduino.

- XBee: ..... Establecen el canal de comunicaciones NODO – COORDINADOR.

- Ordenador: ..... Por un lado, actúa como Módulo "complementario", es decir, si estamos realizando las pruebas del NODO actúa como COORDINADOR y viceversa.

  Por otro lado, mediante el cable de conexión con las placas de Arduino y XBee y el programa X-CTU monitorizamos los datos que pasan por el puerto serie de ambas placas.

Las pruebas del NODO se realizan en dos fases:

- FASE I: ..... Se carga el programa con el comportamiento deseado. Esto se hace una vez conectada la placa al ordenador utilizando el interfaz de desarrollo Arduino IDE.

- FASE II:..... Se conectan los transceptores (XBee) y se comprueba en funcionamiento con el programa X-CTU que monitoriza los datos enviados y recibidos por el puerto serie. Esta comprobación la hemos realizado tanto para la placa Arduino Coordinador como para el Nodo de forma simultánea.

Para **el comportamiento** nos centramos en la siguiente dinámica:

- *El nodo envía el código de sincronismo: /C01N01_ok*

  Inicialmente existe una fase de sincronismo en la que El NODO está continuamente mandando su código hasta que recibe el código del COORDINADOR con el que está asociado.

  El tiempo entre emisiones se fija con la variable "retardo" (inicialmente a 1sg) y es tratada por la rutina "leodatos".

```
if (sinc==0)                    // Si no está sincronizado el nodo
{
  retardo=1000;
  leodatos();·
```

- *El "coordinador" envía el código de sincronismo*

  El simulador del coordinador envía el código de sincronización /C01N01_ok y el Nodo lo recibe

- *El nodo lee las entradas analógicas y envía instrucciones*

  Una vez sincronizados el NODO empieza a leer si hay señales en las entradas.

  Para ello en vamos cambiando la posición de los microinterruptores y vemos que las señales transmitidas son las correctas. En la figura se muestra el resultado de la transmisión

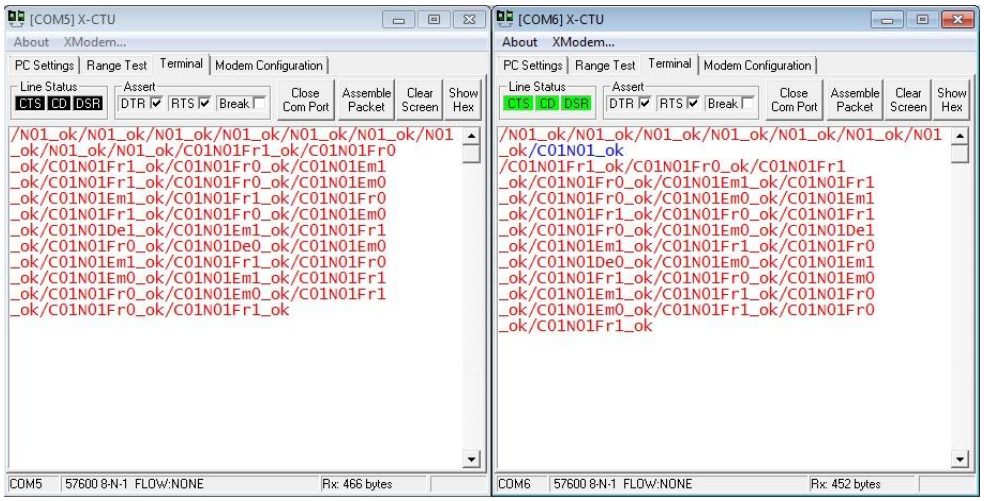


Ilustración 85. Prueba 1 Controlador Nodo[45]

CONTROLADOR COORDINADOR

Para comprobar el correcto funcionamiento del controlador COORDINADOR el banco de pruebas utilizado es:

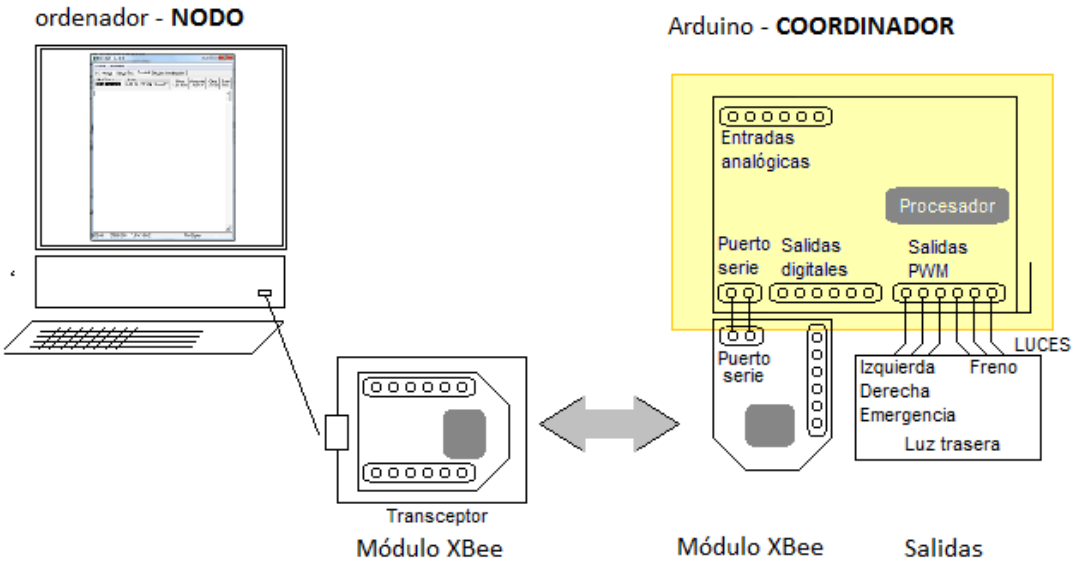


Ilustración 86. Banco de pruebas Arduino Coordinador.

[45] Debido a la versión del programa en el momento de la captura de la imagen, el código enviado para sincronismo es /C01N01_ok en lugar de /C01N01Si1_ok como se ha definido en la memoria.

Los elementos auxiliares utilizados para las pruebas del Arduino COORDINADOR según se ha indicado en la "Ilustración 86. Banco de pruebas Arduino Coordinador." son:

- Salidas: ..... Para las salidas utilizamos leds que nos permiten monitorizar el estado de encendido / apagado de las señales (1 o 0).
- XBee: ..... Establecen el canal de comunicaciones NODO – COORDINADOR.
- Ordenador: ..... Por un lado, actúa como Módulo "complementario", es decir, si estamos realizando las pruebas del COORDINADOR actúa como NODO y viceversa.

  Por otro lado, mediante el cable de conexión con las placas de Arduino y XBee y el programa X-CTU monitorizamos los datos que pasan por el puerto serie de ambas placas.

Las pruebas del COORDINADOR se realizan en dos fases:

- FASE I: ..... Se carga el programa con el comportamiento deseado. Esto se hace una vez conectada la placa al ordenador utilizando el interfaz de desarrollo Arduino IDE.

  Para comprobar el funcionamiento, usando la monitorización del puerto COM4 desde Arduino IDE:

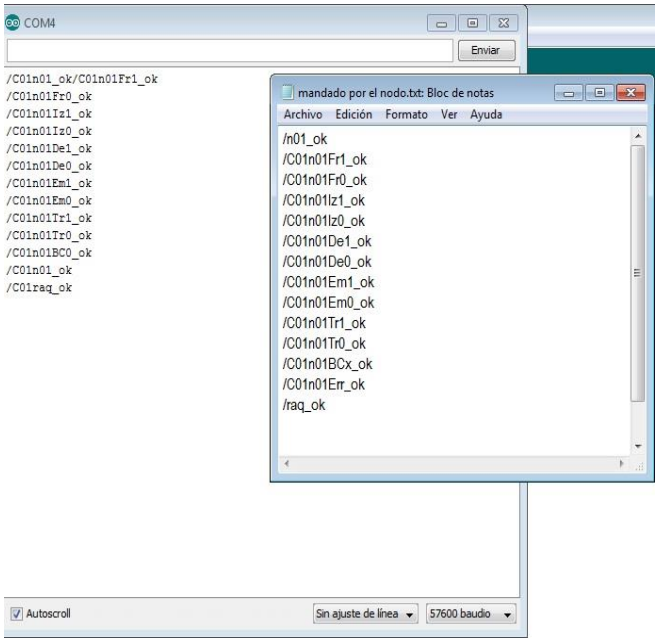


Ilustración 87. COM4 – Arduino IDE[46]

[46] En la imagen aparecen códigos que se incluyeron en la programación exclusivamente para facilitar la monitorización y la realización de las pruebas, como es el caso de "raq_ok"

- FASE II:..... Se conectan los transceptores (XBee) y se comprueba en funcionamiento con el programa X-CTU que monitoriza los datos enviados y recibidos por el puerto serie. Esta comprobación la hemos realizado tanto para la placa Arduino Coordinador como para el Nodo de forma simultánea.

Para **el comportamiento** nos centramos en la siguiente dinámica:

- *"Escucha activa"*

  El COORDINADOR está continuamente "escuchando" hasta que recibe un "código orden" del NODO con el que está asociado (/C01N01XYZ_ok).

- *"Recepción de orden"*

  Cuando se recibe un código de orden, se procesa mediante el programa generado a tal efecto y si procede, opera con la señal a la que se refiere la orden (la enciende o la apaga).

- "*Envío de confirmación de orden ejecutada correctamente*"

  Por último, una vez ejecutada la acción envía al NODO una confirmación.

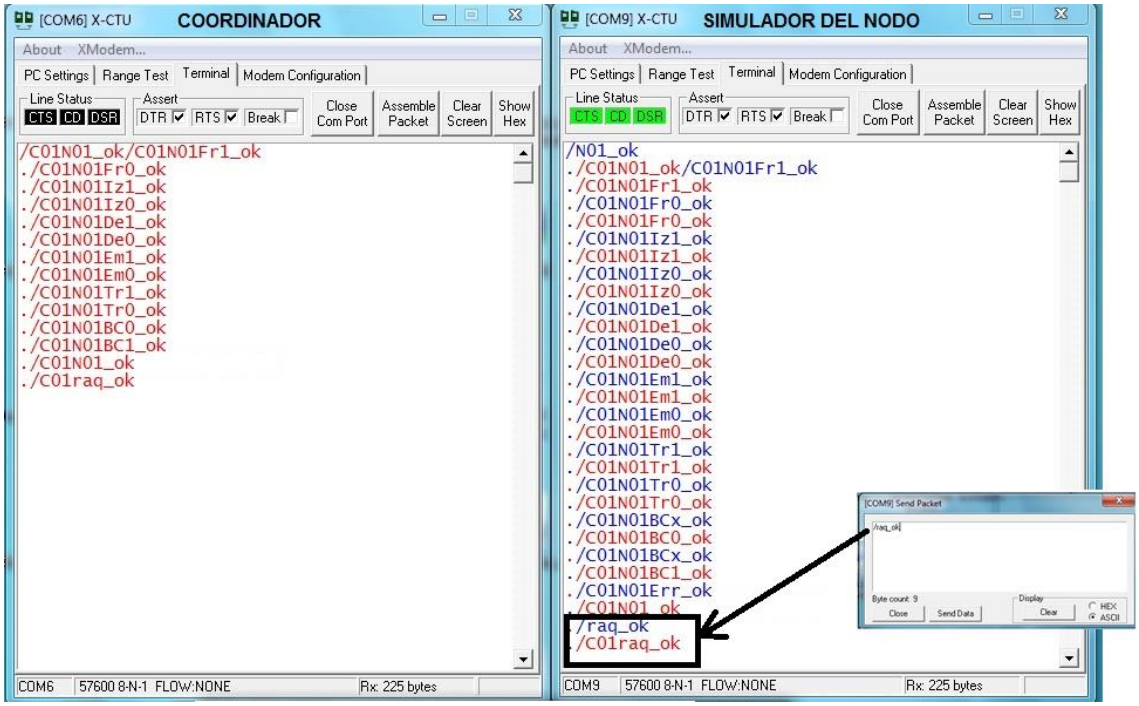


Ilustración 88. Simulación XCTU Nodo - Coordinador

7 ANÁLISIS DE VIABILIDAD

El objetivo de este punto es presentar los costes asociados al desarrollo del proyecto y analizarlos brevemente.

Entendemos como **costes totales** del proyecto son la suma de "*costes de producto*" y "*costes de desarrollo*"

Como adelantamos en la tabla a continuación, el coste del producto es muy inferior al coste de desarrollo por lo que consideramos **coste total ≈ coste de desarrollo**.

Coste de producto	Coste de desarrollo	Coste total
183,30 €	13.825,00 €	14.008,30

Tabla 42. Resumen de costes

Si bien no se analizan costes en los que no se ha incurrido, como aquellos derivados de la producción (diseño de prototipos, moldes, almacenaje, viajes…), de Marketing y Comercialización de producto o asociados a impuestos y aranceles, podemos realizar una **estimación del comportamiento** tanto de los costes como de los beneficios esperados.

- **Coste de desarrollo**: Es un coste fijo que se realiza en la fase de inicio / desarrollo del proyecto.
- **Coste de producto:** Nos referimos al coste unitario de cada pieza. Inicialmente, su relevancia es baja dado que los costes de desarrollo son muy superiores. A medida que aumenta el número de unidades vendidas observamos:
  - Aumenta su importancia respecto al coste total, ya que los gastos de desarrollo en un intervalo de tiempo permanecen constantes, mientras que a mayor número de unidades fabricadas, mayor inversión y coste total.
  - Baja el coste de producto unitario: en el mercado el precio unitario es menor a mayor número de unidades adquiridas.
- **Beneficio** = Precio de venta – (Coste de producto + amortización de coste de desarrollo + otros costes)

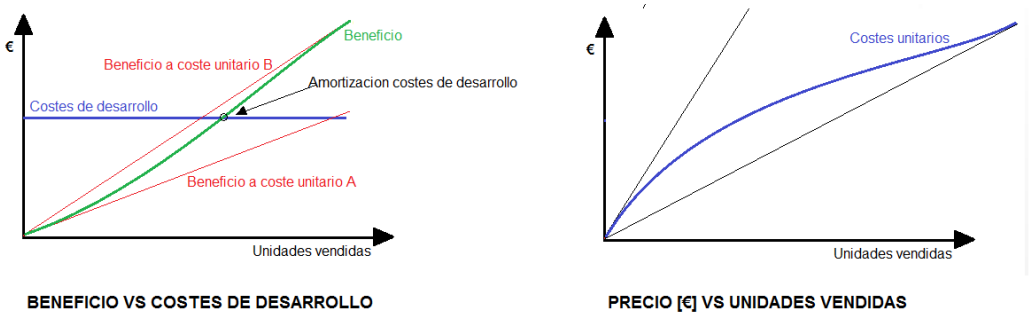


Ilustración 89. Beneficio, Coste de desarrollo y Coste unitario

7.1 PRESUPUESTO

Para la realización del proyecto se han empleado los siguientes recursos:

1. EQUIPAMIENTO.

Concepto	Precio €	% Dedicación al proyecto	Precio total
Ordenador Portátil	790,00 €	50,00%	395,00 €
Fuente de alimentación	30,00 €	100,00%	30,00 €
Osciloscopio de mano DSO100	160,00 €	100,00%	160,00 €
Polímetro	20,00 €	50,00%	10,00 €
Soldador	30,00 €	100,00%	30,00 €
TOTAL			**625,00 €**

En el gráfico siguiente se muestra el peso de cada concepto sobre el total del equipamiento:

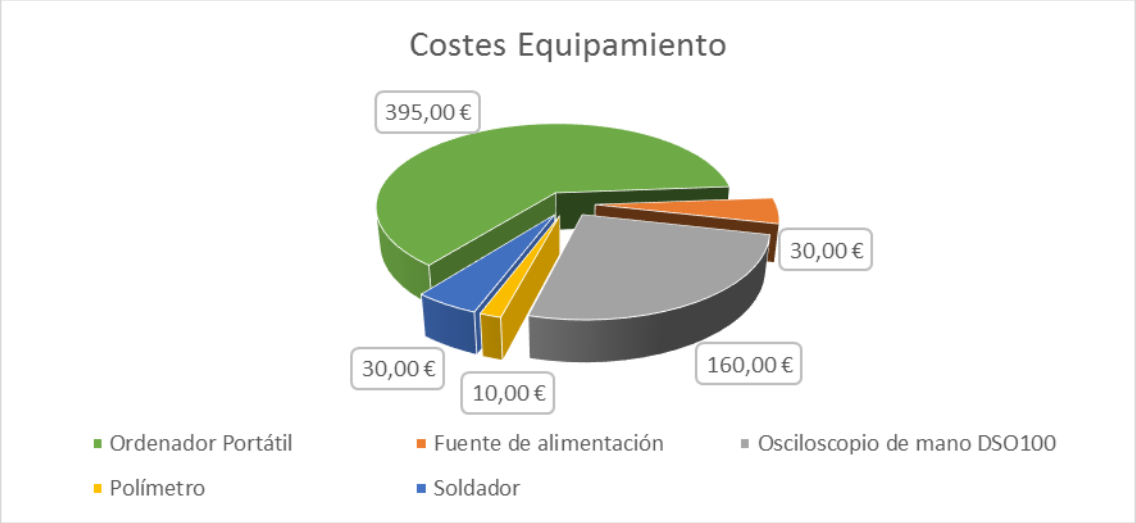


Ilustración 90. Coste Equipamiento.

2. COMPONENTES.

Concepto 1	Descripción	Precio €	Unidades	Precio total
Transmisor/Receptor	*X-Bee*	25,00 €	2	50,00 €
SHIELD XBee	*Adaptador Xbee Arduino*	13,14 €	2	26,28 €
Controlador	*Arduino UNO*	19,01 €	2	38,02 €
Tira LED RGB (1m)	*LED RGB para freno*	19,00 €	1	19,00 €
Juego intermitentes	*Módulos Led de intermitentes*	5,00 €	1	5,00 €
Bateria 9V	*Baterías recargables*	10,00 €	2	20,00 €
Otro material	*Cables, componentes, interruptores y pulsadores.*	25,00 €	1	25,00 €
TOTAL				183,30 €

En el gráfico siguiente se muestra el peso de cada concepto sobre el total de Componentes:

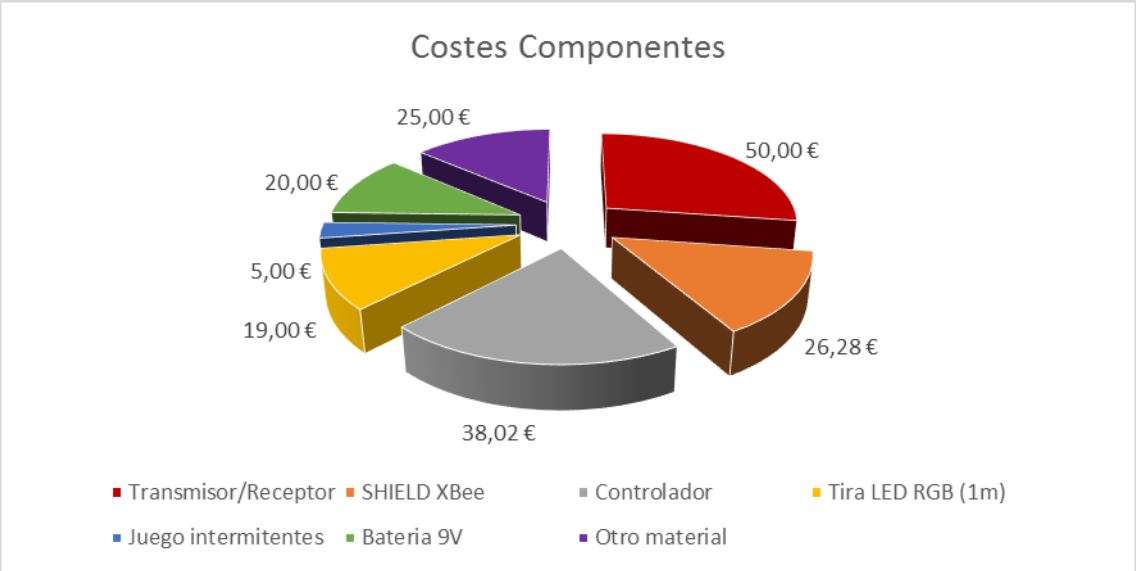


Ilustración 91. Coste Componentes.

3. RECURSOS HUMANOS.

Para el cálculo de los costes de personal se supone un coste bruto hora de 30,00€ para un Ingeniero Técnico Industrial.

Concepto 1	Descripción	Precio unidad	Unidades	Precio total
Formación inicial	*Formación en Arduino y XBee y pruebas iniciales*	30,00 €	80	2.400,00 €
Diseño y montaje	*Elección de componentes y montaje del hardware*	30,00 €	40	1.200,00 €
Programacion	*Programación de XBee y Arduino*	30,00 €	190	5.700,00 €
Documentacion	*Proyecto y programas*	30,00 €	130	3.900,00 €
Otros trabajos	*Compras, Pruebas y validación, etc.*	30,00 €	80	2.400,00 €
TOTAL				**15.600,00 €**

En el gráfico siguiente se muestra el peso de cada concepto sobre el total de Componentes:

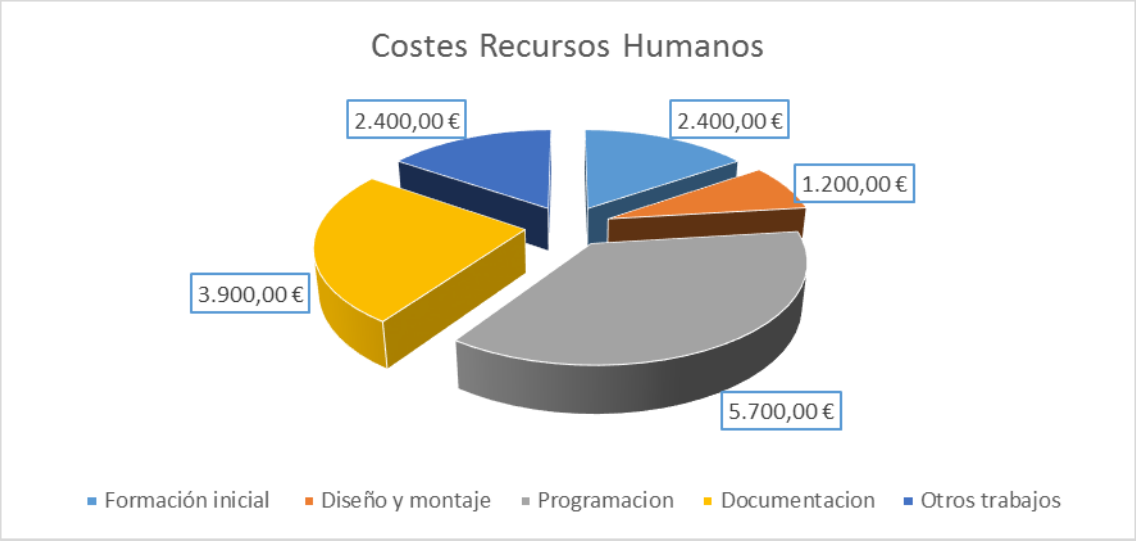


Ilustración 92. Coste Recursos Humanos.

8 CONCLUSIONES Y TRABAJOS FUTUROS

La reciente aparición de productos sustitutivos que aportan funcionalidades similares, unido al aumento progresivo en los últimos años de los usuarios de las bicicletas y los esfuerzos de las Administraciones por acondicionar las vías y aumentar la seguridad de los ciclistas indican que actualmente existe una necesidad a cubrir.

Para ello, la tecnología actual se encuentra en un estado de desarrollo que nos permite conformar un producto mediante productos existentes en el mercado, como los módulos XBee y las placas Arduino,

Además, destaca como punto fuerte del sistema que es ampliamente escalable, y existen múltiples mejoras y aplicaciones que se pueden implementar con sencillos desarrollos o incluso, simplemente mediante una modificación en la programación de los módulos de control, sin inversiones adicionales en Hardware ni rediseño de circuitos.

Respecto a la comercialización del producto, si bien es cierto que es necesario realizar un estudio de mercado para conocer el precio objetivo para el segmento al que se oriente la comercialización del producto, y un trabajo importante en la optimización de costes con los datos actuales entendemos que:

- Con los costes de desarrollo y producto actuales, el precio del producto resulta elevado para las expectativas del mercado y por tanto su comercialización no es viable.

Sin embargo, hemos estimado que los costes de producto pueden bajar hasta situarse en menos de un tercio de los costes actuales, momento en el que su fabricación y comercialización se entendería viable de cara a obtener beneficios.

A continuación, se adjuntan una serie de propuestas de evolución del producto mediante integraciones o re-programaciones que se ha considerado aportarán valor añadido, dejándose planteadas para desarrollos futuros.

	Descripción
1	Mejoras en la monitorización en el NODO
2	Mejora en la señalización del Freno
3	Añadir sensor de luz
4	Añadir luces de posición
5	Adaptación a sistema Coordinador - Multinodo
6	Nuevos elementos: Retrovisores con intermitentes
7	Módulos de recarga de batería
8	Optimización de Hardware

Tabla 43. Trabajos futuros.

1. Mejoras en la monitorización en el NODO

Monitorización del estado de la batería del coordinador e integración con el aviso actual de batería baja del NODO.

2. Mejora en la señalización del Freno: Acelerómetro

Sustituyendo el pulsador de freno por un acelerómetro integrado en el sistema, no sólo se indica mediante señales luminosas que se pierde velocidad con la pulsación sino también en situaciones como los cambios bruscos de pendiente, agotamiento del usuario.

3. Añadir sensor de luz

a. Variación de la intensidad lumínica de los LEDs del panel de señalización a su punto óptimo de luminosidad en función de la luz exterior reduce consumo de energía.

b. En el caso de integrar luces de posición, encendido y apagado automático en función de la normativa (salida y ocaso del sol, túneles, etc) de forma transparente para el usuario.

4. **Añadir luces de posición:**

Integrar elementos obligatorios como las luces de posición delantera y trasera de forma que toda la iluminación del vehículo está integrada en un dispositivo único.

5. Adaptar las funcionalidades a un sistema Coordinador - Multinodo

Manteniendo las funcionalidades adaptar el trabajo realizado a un sistema en el que un dispositivo COORDINADOR controle simultáneamente multitud de NODOS. De esta forma el vehículo que lleva el módulo principal envía una orden que se ejecuta en todos los nodos de su red.

Una aplicación directa de esta funcionalidad es, por ejemplo, la circulación de grupos; Visitas guiadas en bicicleta, segway, etc. En las que el guía realiza la pulsación y la señal de giro se activa en todos los vehículos.

6. Integración de nuevos elementos para la señalización

En el estudio realizado en el punto “3.1.2 Productos sustitutivos” se presentaron elementos como un manillar y un retrovisor con luces integradas.

Estos elementos se podrían incluir en el sistema como nodos adicionales, de forma que no solo aportasen las ventajas que se indican en la Tabla 35, sino que además 3servirían como panel de monitorización de las intermitencias, sustituyendo a los leds ubicados en el NODO para tal efecto.

	Ventajas	**Productos similares**
Manillar	Incluir un módulo receptor en el manillar (equidistantes y elementos más externos del vehículo) aumenta la visibilidad y por tanto la seguridad del usuario.	Blinkergrips[47] 

[47] http://blinkergrips.com/

Retrovisores	Permite al usuario no sólo ser más visible, sino tener un mayor control sobre el resto de vehículos mediante el aporte de retrovisores.	Winkuu[48] 

Tabla 44 Nuevos elementos propuestos a integrar.

7. Células fotovoltaicas para recarga automática de baterías.

El hecho de disponer de baterías con sistema de recarga universal (micro usb, etc.) nos permite la incorporación inmediata de elementos externos de recarga de baterías incluso simultáneamente a su utilización por parte del usuario.

No obstante, podría realizarse la integración de un módulo de carga, por ejemplo mediante células fotovoltaicas.

8. Optimización de Hardware: Arduino FINO

Estudio de la migración de la placa de Arduino UNO a Arduino FINO, ya que este último lleva integrado el módulo Xbee.

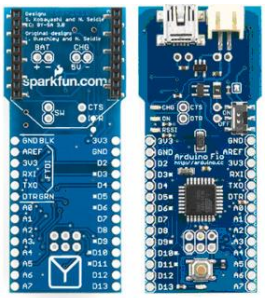


Tabla 45. Arduino FINO

[48] http://www.winkku.co.uk/three.php

BIBLIOGRAFÍA

[Internet] Estudio sobre las estrategias de promoción de la bicicleta como medio de transporte en las ciudades Españolas – BACC para la DGT
http://www.dgt.es/Galerias/seguridad-vial/investigacion/estudios-e-informes/INFORME-ESTUDIO-SOBRE-LAS-ESTRATEGIAS-DE-PROMOCION-DE-LA-BICICLETA-9.pdf

[Internet] - Acondicionamiento de señales – Instrumentación Electrónica.
http://ocw.uc3m.es/tecnologia-electronica/instrumentacion-electronica-i/

[Internet] Sistemas Digitales de Instrumentación y control
http://upcommons.upc.edu/e-prints/bitstream/2117/6121/1/TEMA1.pdf

[Internet] Acondicionamiento de señales
http://informatica.uv.es/iiguia/INS/material/inst02.pdf
http://www.ni.com/white-paper/4084/es/
http://1538445.blogspot.com.es/2012/11/22-acondicionamiento-de-senales.html

[Internet] Apuntes Electrónica Analógica
http://blog.educastur.es/tecnoaller/files/2011/02/apuntes-e-analogica.pdf

[Internet] Apuntes UPV – PICs y Sensores
http://server-die.alc.upv.es/asignaturas/PAEEES/

[Internet] Sensores y Acondicionadores
http://es.slideshare.net/astranegro/14054883-transductoressensores
http://es.slideshare.net/mdovale/sensores-y-acondicionadores

[Internet] Apuntes UVA – Microprocesadores.
http://www.infor.uva.es/~fernando/personal/

[Internet] Libro electrónico: El Mundo de los Microcontroladores
http://www.mikroe.com/chapters/view/84/libro-de-la-programacion-de-los-microcontroladores-pic-en-basic-capitulo-1-mundo-de-los-microcontroladores/

[Internet] Información Control - PICs
http://es.wikipedia.org/wiki/Microcontrolador_PIC#PIC_modernos

[Internet] Apuntes microcontroladores PIC
http://perso.wanadoo.es/pictob/micropic.htm

[Internet] Optoelectrónica
http://www.geocities.ws/curso_tecnologia_electronica/TemasTE2/TE2-T05C.pdf

[Internet] Fotoemisores
http://server-die.alc.upv.es/asignaturas/lsed/2002-03/Sensores_Luz/fotoemisores.htm

[Internet] Tutoriales varios – Puertas lógicas
http://www.tutoelectro.com/tutoriales/electronica-basica/puertas-logicas/

[Internet] Websites con proyectos de electrónica
http://www.pickey.es/control-de-reles-a-distancia-por-radiofrecuencia--rf-.html
http://www.forosdeelectronica.com

[Internet] Apuntes Tecnología de la comunicación
http://www.edu.xunta.es/centros/iesfelixmuriel/system/files/Tecno_comunicacion.pdf

[Internet] Apuntes Introducción a Sistemas de Telecomunicación.
http://agamenon.tsc.uah.es/Asignaturas/ie/sis_com/apuntes

[Internet] Productos sustitutivos – Página de Internet de venta de productos.
http://www.banggood.com

[Internet] Venta de Hilo Conductor.
http://www.reflexiona.biz/shop/materiales/285--hilo-conductivo-117-17.html

[Internet] Arduino – Página oficial
https://www.arduino.cc

[Internet] Arduino – "The Documentary"
https://vimeo.com/18390711

[Internet] Fritzing
http://Fritzing.org/home.

[Internet] Arduino – Otras fuentes
http://joautomation.com/wp-content/uploads/2015/03/arduino_uno_pinout.png
http://www.arduineando.com/tutoriales/
http://www.educachip.com/alimentar-arduino/
http://aitormartin-apuntes.blogspot.com.es/2013/12/arduino-varios-pulsadores-por-linea-de.html
http://arduino-guay.blogspot.com.es/p/lo-mejor-sobre-arduino.html

[Internet] Arduino – Comunicación Puerto Serie
http://www.luisllamas.es/2014/04/arduino-puerto-serie/

[Internet] Arduino – Pulsadores
http://aitormartin-apuntes.blogspot.com.es/
http://www.diarioelectronicohoy.com/blog/pulsadores-sin-rebotes

[Internet] Arduino – Información de Shield XBee.
http://arduino.cc/es/Main/ArduinoXbeeShield#.U1jSOhybvzT
http://arduino.cc/es/Guide/ArduinoXbeeShield#.U1jSBxybvzR

[Internet] Portal didáctico - Integración de módulos Arduino y Xbee
http://www.andresduarte.com/arduino-y-xbee
http://fuenteabierta.teubi.co/2014/03/arduino-y-el-xbee-series-1-modo-api.html

[Internet] X-Bee
www.digi.com/lp/xbee/

[Internet] X-Bee Guía de Usuario.
www.olimex.cl/pdf/Wireless/ZigBee/XBee-Guia_Usuario.pdf

[Internet] X-Bee – Guía de referencia rápida
http://www.tunnelsup.com/images/XBee-Quick-Reference-Guide.pdf

Otros
www.wikipedia.es

[Internet] Proyectos personales relacionados: cómo hacer luz de giro para bicicleta
http://www.taringa.net/post/hazlo-tu-mismo/14728711/
http://comunidad.dragonjar.org/f210/luz-de-giro-para-bicicleta-7998/
http://www.tiempogeek.com/2013/04/11/senalizacion-ropa-electronica/

WEBSITES CON PROPUESTAS A INTEGRAR

[Internet] Arduino – Medir nivel de luz con Arduino y LDR
http://www.luisllamas.es/2015/03/medir-nivel-luz-con-arduino-y-fotoresistencia-ldr/

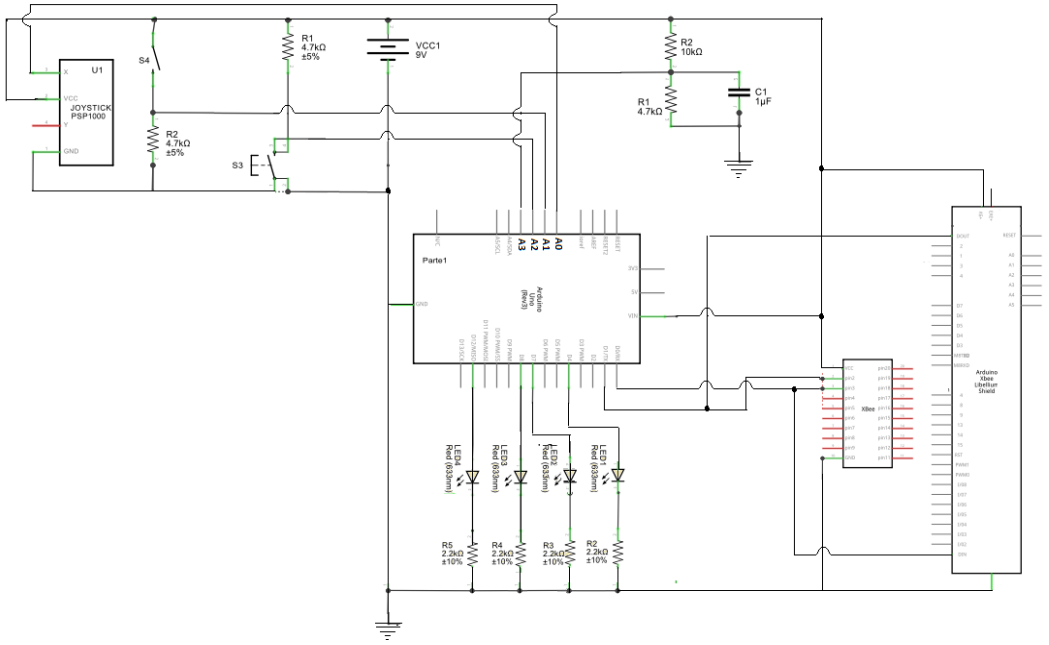
[Internet] Arduino – Proyecto sensor temperatura inalámbrico RF.
http://www.hell-desk.com/comunicacion-inalambrica-arduino-nrf24l01/

[Internet] Arduino - Microinterruptor
http://www.trebol-a.com/2015/05/28/microinterruptor-para-arduino/

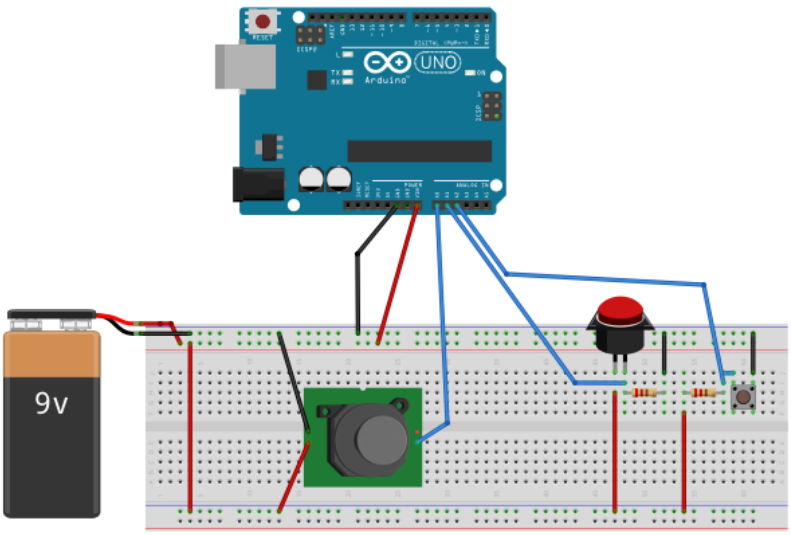
[Internet] Arduino - IR
http://electronicavm.wordpress.com

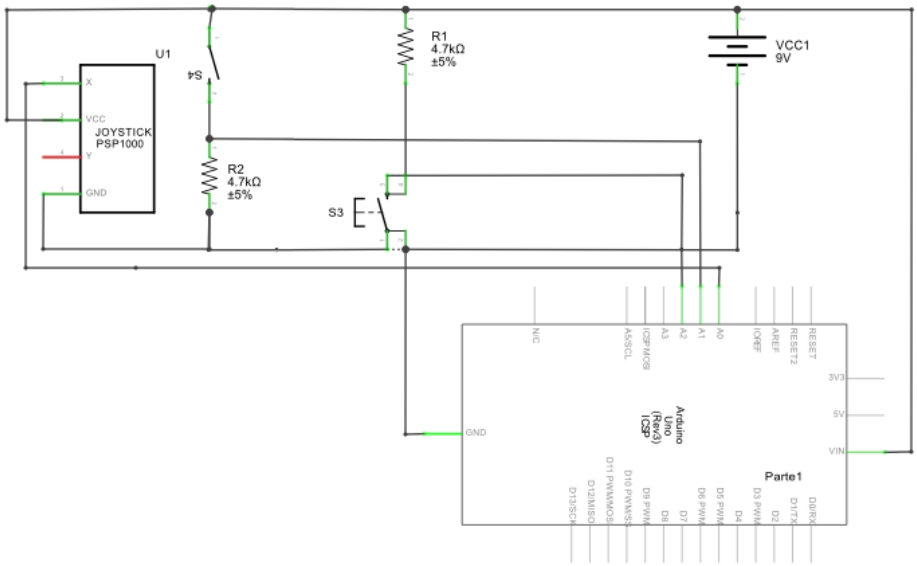
9 ANEXO I. PLANOS Y HOJAS DE CARACTERÍSTICAS

9.1 PLANOS

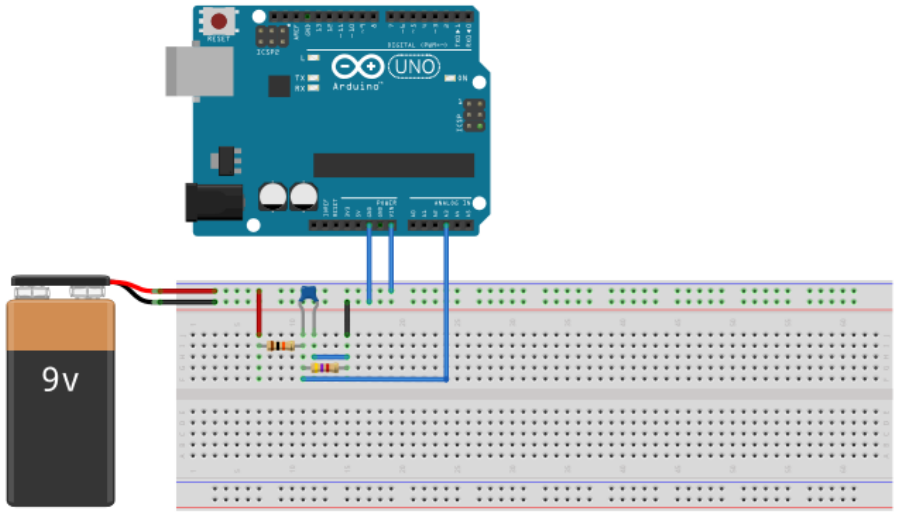


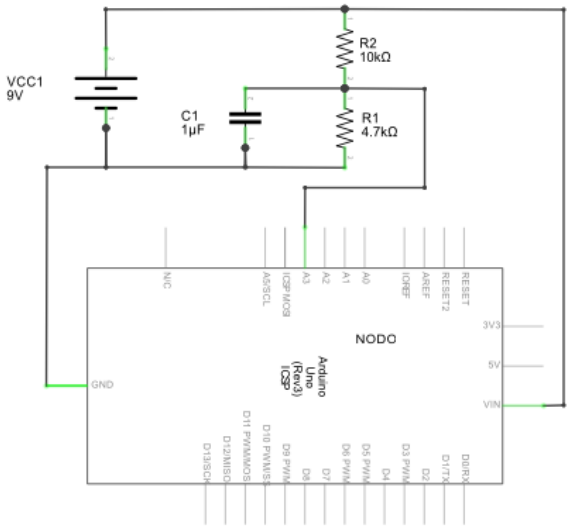
Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
NODO – SISTEMA COMPLETO		NODO ENTRADAS	N01



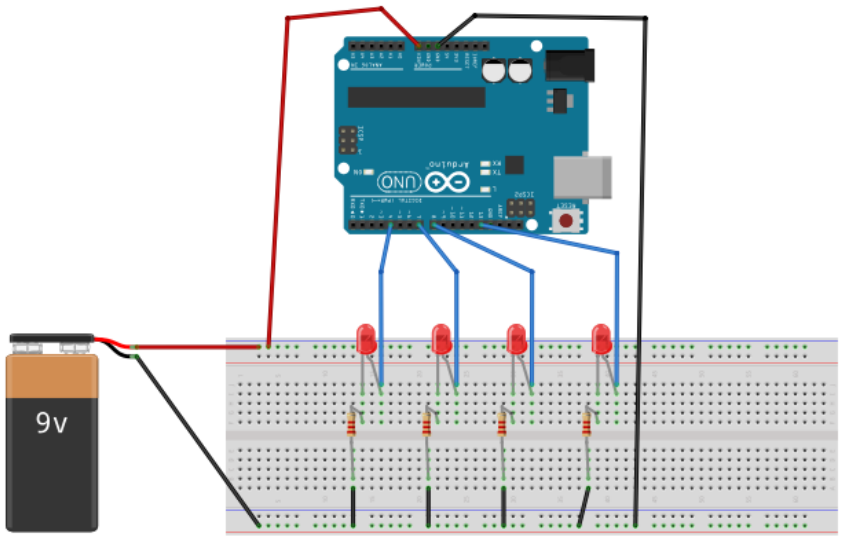


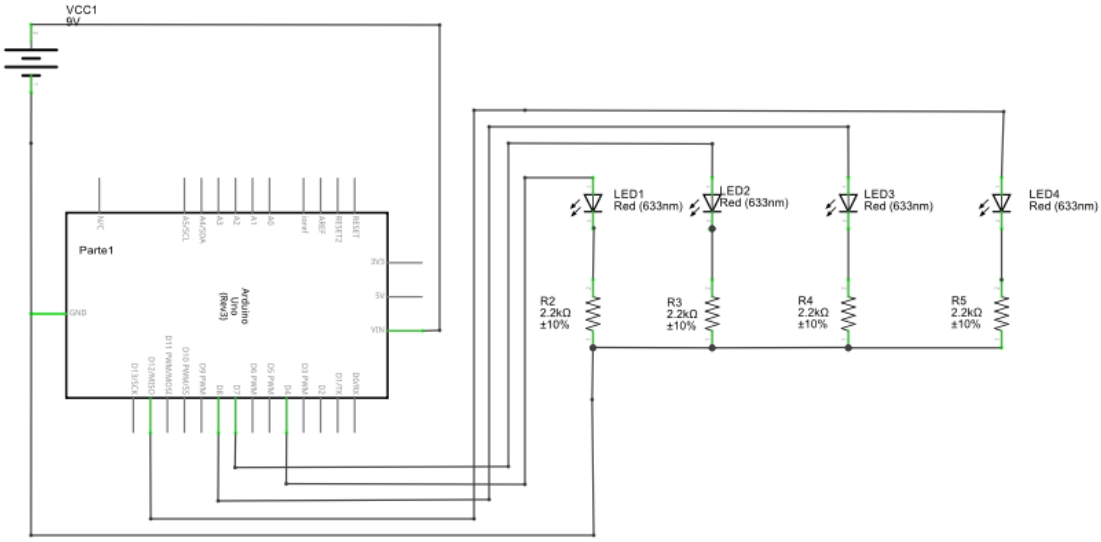
Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
CONEXIÓN ARDUINO: FRENO, EMERGENCIA, INTERMITENTES.		NODO ENTRADAS	N02



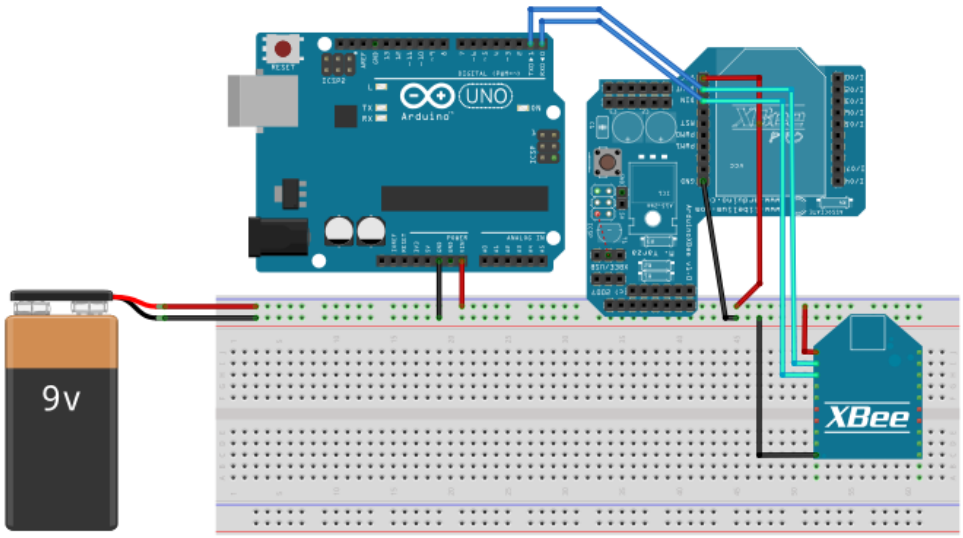


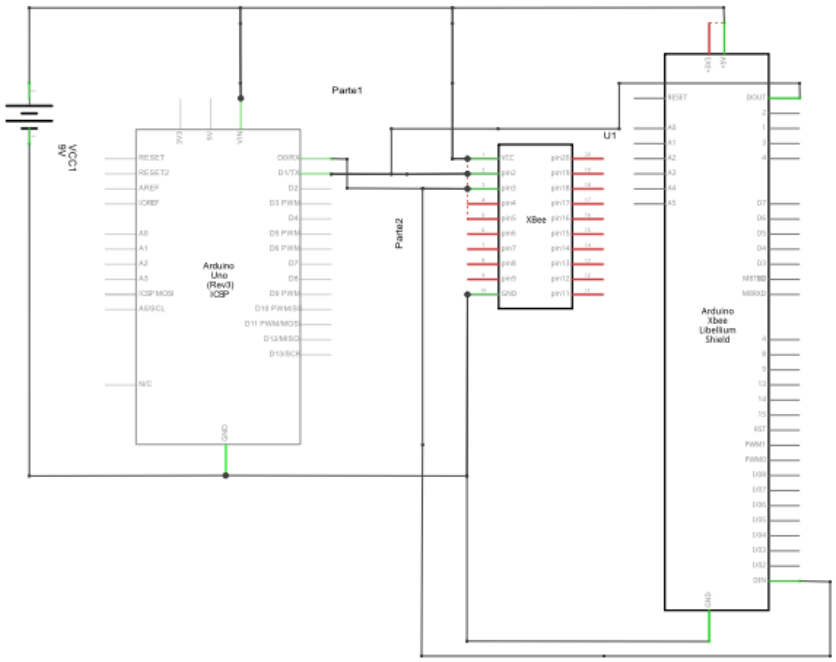
Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
CONEXIÓN ARDUINO MONITORIZACIÓN BATERÍA		NODO ENTRADAS	N03



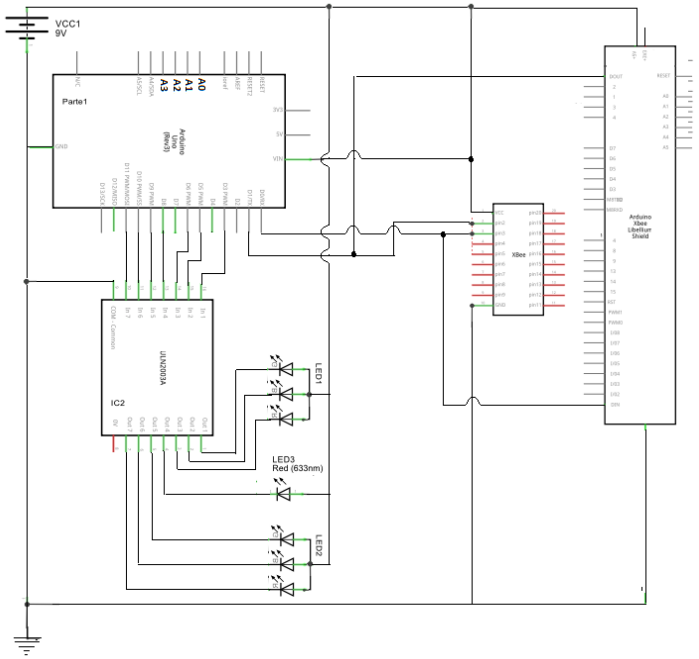


Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
CONEXIÓN ARDUINO LEDS MONITORIZACIÓN		NODO SALIDAS	N04

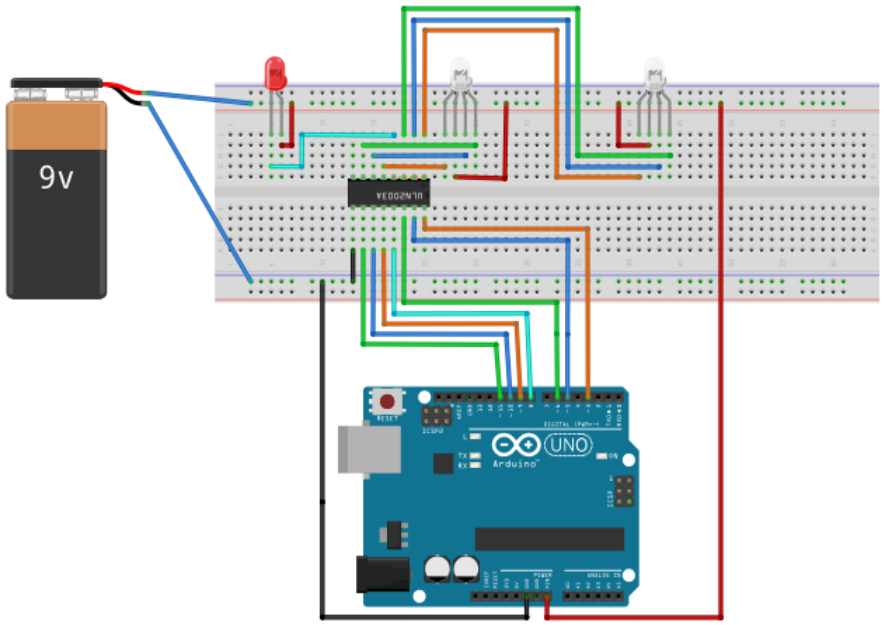


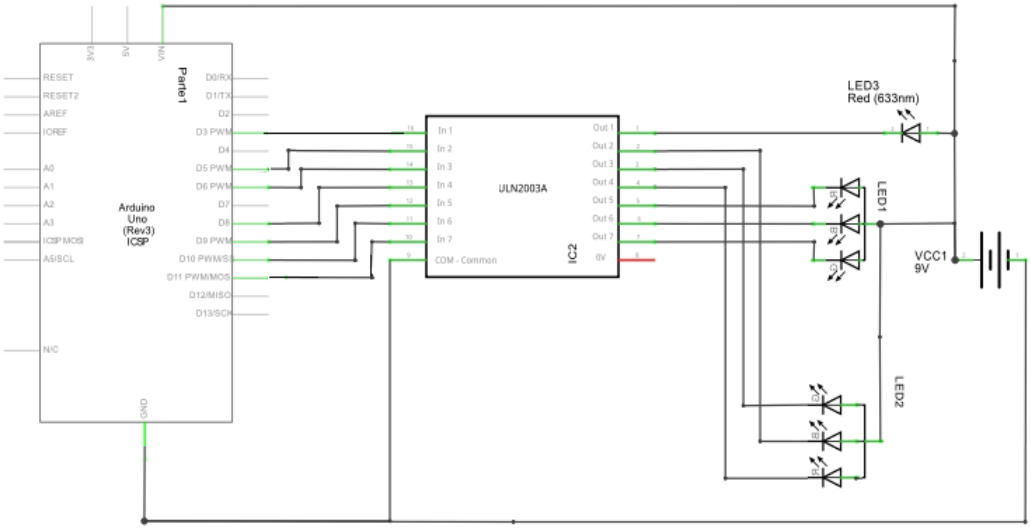


Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
CONEXIÓN ARDUINO - XBEE		NODO SALIDAS	N05



Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
NODO – SISTEMA COMPLETO		NODO ENTRADAS	N01





Fecha	01/09/2015	PFC - SISTEMA DE SEÑALIZACIÓN PARA CICLISTAS	
Nombre y Apellidos	Ruiz Cañamero, Raquel		
DENOMINACIÓN DEL PLANO		GRUPO	PLANO Nº
CONEXIÓN SALIDAS PWM CON REGULACIÓN DE COLOR EN INTERMITENTES.		COORDINADOR SALIDAS	C01

10 ANEXO II.SOFTWARE.

10.1 CODIGO FUENTE COORDINADOR

// VARIABLES GLOBALES

```
//  VARIABLES ASOCIADAS AL HARDWARE DEL CONTROLADOR
int salidadig[6]={2,4,7,8,12,13};                    // Array que contiene los pines Digitales
int salidapwm[6]={3,5,6,9,10,11};                    // Array que contiene los pines PWM
int contenidoanalog[6];                 // Vector de los valores de las entradas analógicas
int conenido_A0;
int conenido_A1;
int conenido_A2;
int conenido_A3;
int limite_bat;                 // Evita rebotes en el encendido de la alarma por batería baja
char codigo_global[6]={'C','0','1','N','0','1'};        // Variable que contiene el código global
char codigo_fin[3]={'_','o','k'};                // Contiene los caracteres de fin de información

//  VARIABLES ASOCIADAS A LAS ORDENES RECIBIDAS DEL NODO
int sinc;                                    // Variable que indica si existe sincronizacion
int izquierda;                          // Actuación del intermitente izquierdo en el NODO
int derecha;                            // Actuación del intermitente derecho en el NODO
int emergencia;                                 // Actuación dela emergencia en el NODO
int freno;                          // Variable asociada a la actuación del freno en el NODO
int bateria;                        // Variable que indica el estado de la batería en el NODO
int izquierda_to;                       // Valor de la variable izquierda en el ciclo anterior
int derecha_to;                         // Valor de la variable derecha en el ciclo anterior
int emergencia_to;                  // Valor de la variable emergencia en el ciclo anterior
int freno_to;                           // Valor de la variable freno en el ciclo anterior
int bateria_to;                         // Valor de la variable batería en el ciclo anterior
int cambio;                         // Cambio en los valores de las entradas en el NODO
char codigo_orden[3];               // Variable que contiene la orden recibida/a transmitir
char codigo_orden_enviado[3];                   // Variable que contiene la orden enviada
char codigo_orden_recibido[3];                  // Variable que contiene la orden recibida

//  VARIABLES ASOCIADAS A LA RUTINA CÍCLICA (loop)
int error;                                  // Variable asociada a errores en el proceso
int i;

//  VARIABLES ASOCIADAS CON TEMPORIZADORES
int ciclo_emergencia;
int ciclo_derecha;
int ciclo_izquierda;
float to;                                   // Almacena el tiempo en el que se ejecuta un ciclo
float to_ant;                           // Almacena el tiempo de ejecución del ciclo anterior
float t_fin;                                    // Apagado automático de señalización
float t_sinc;                                   // Emisión del código de sincronismo
float t_lat;                                            // Latido del COORDINADOR
```

```
float temporizador_orden;                    // temporizador en la ejecución de una orden
float inicio_cont;                           // Variables para generar un temporizador
float cont;                                  // Variables para generar un temporizador
float t_emer;                                // evitar rebotes en el intermitente derecho
float t_de;                                  // evitar rebotes en el intermitente derecho
float t_iz;                                  // evitar rebotes en el intermitente izquierdo

// VARIABLES ASOCIADAS A ESCRIBIR POR EL PUERTOSERIE

int inter_iz;                        // Actuación intermitente izquierdo en el COORDINADOR
int inter_de;                        // Actuación intermitente derecho en el COORDINADOR
int inter_em;                        // Actuación luces de emergencia en el OORDINADOR
int he_escrito_datos;                // Escritura de un bloque de datos por el puerto serie
int escribir;                                // Escritura de datos por el puerto serie
int error_escribir;                  // Escritura de un bloque de datos por el puerto serie

// VARIABLES ASOCIADAS A LA LECTURA DE DATOS POR EL PUERTO SERIE
int he_leido_datos;                  // lectura de un bloque de datos por el puerto serie
```

// CONFIGURACIÓN SETUP

```
void setup()
{
//   LA CONFIGURACIÓN DEL CONTROLADOR COORDINADOR
//   A continuación se recojen el conjunto de instrucciones que permiten
//   configurar el Hardware del controlador COORDINADOR:
//   1.- Configurar las salidas Digitales
//   2.- Configurar las salidas PWM
//   3.- Configurar el Puerto Serie

        for (i=0;i<6;i++)
                {
                        pinMode(salidadig[i],OUTPUT);
                        pinMode(salidapwm[i],OUTPUT);
                }
        Serial.begin(57600);

//        CONFIGURACIÓN INICIAL DE VARIABLES
//   A continuación se recojen el conjunto de variables que han de ser pre-
//   configuradas antes de iniciar el programa cíclico
//   1.- Sincronismo
//   2.- Temporizador de latido
//   3.- Temporizador de sobre tiempo

        for (i=0;i<6;i++)
                {
                        digitalWrite(salidadig[i],LOW);
                        analogWrite(salidapwm[i],0);
 }
 sinc=0;
 freno=0;
 emergencia=0;
 derecha=0;
 izquierda=0;
```

```
 freno_to=0;
 emergencia_to=0;
 derecha_to=0;
 izquierda_to=0;
ciclo_emergencia=0;
 ciclo_derecha=0;
 ciclo_izquierda=0;
 limite_bat=650;
 t_lat=(millis()/1000)+60;
 to_ant=millis();
t_fin=(millis()/1000)+40;
 t_sinc=(millis()/1000)+10;
}

// PROGRAMA CICLICO
void loop()
{
//        ACCIONES AL COMENZAR UN CICLO DE PROGRAMA
//   Antes de comenzar cada ciclo de programa haya que realizar una serie de
//   acciones:
//   1.- iniciar las variables globales
//   2.- Definir las variables que se utilizarán en la Rutina
//   3.- Leer el tiempo en la que se ejecuta el ciclo
//   4.- Corregir errores en lectura de tiempo

error=0;
error_escribir=0;
to=millis();
if (to < to_ant)
        {
                t_lat=(to/1000)+40;
                t_fin=(to/1000)+20;
                t_sinc=(to/1000)+10;
        }
 else
 {
                to_ant=to;
 }

//                  PROGRAMA CÍCLICO DEL COORDINADOR
//   Si está sincronicado, el COORDINADOR ha de realizar cíclicamente:
//   1.- Sincronizarse y garantizar el sincronismo (latido)
//   2.- Leer las órdenes que llegan por el puerto serie del NODO
//   3.- Si llegan, convertir las ordenes en acciones sobre las luces
//   4.- Escribir por el puerto serie la confirmación de orden recibida

if(sinc==0)
        {
                codigo_orden[0]='S';
                codigo_orden[1]='i';
                codigo_orden[2]='1';
                leer_puertoserie();
                if(he_leido_datos >0 && error==0 && sinc==1)
                        {
```

```
                                escribir=1;
                                escribir_puertoserie();
                        }
        }
  if (sinc==1)
        {
                error=0;
                leer_puertoserie();
    if (he_leido_datos==1 && error==0)
                        {
                                interpretar_orden();
                                ejecutar_orden();
                                escribir=1;
                                escribir_puertoserie();
                        }
                else
                        {
                                interpretar_orden();
                                ejecutar_orden();
                        }
        }

//      ACCIONES AL TERMINAR UN CICLO DE PROGRAMA
//  Al terminar un ciclo de programa hay que comprobar temporizadores
//  5.- Enviar latido

        if ((to/1000)>t_lat && sinc==1)
                {
                        codigo_orden[0]='L';
                        codigo_orden[1]='a';
                        codigo_orden[2]='t';
                        escribir=1;
                        escribir_puertoserie();
                        t_lat=(to/1000+40);
                }
}
```

```
// RUTINA DE LECTURA DEL PUERTO SERIE
//   Esta rutina realiza la lectura del puerto serie del controlador.
// 1.- Defino variables locales
// 2.- Inicializo variables
// 3.- Lee el bloque de información recibido
//   3.1.- Longitud fija de 13 caracteres
//   3.2.- Detecta el inicio de bloque de información
//   3.3.- Autentica con código global asignado a COORDINADOR-NODO
//   3.4.- Extrae la orden recibida
//   3.5.- Comprueba que es una orden válida
// 4.- Detecta la señal de sincronismo
// 5.- Detecta la señal de latido
// 6.- Deja registros de lectura de datos y de errores

void leer_puertoserie()
{
  int n_total_leidos;  // El  numero de caracteres recibidos
          int n_leidos=0;           // El numero de caracteres por leer
          int error_autenticacion=0;// Error en la autenticación del bloque
  int error_fin_bloque=0; // El fin de bloque no es correcto
 char dato[12];             // variable con caracteres recibidos válidos
  char basura[1];           // variable para borrar datos no válidos
  error=0;
  he_leido_datos=0;
  for (i=0;i<12;i++)
    {
        dato[i]=' ';
    }
  n_leidos=Serial.available();
  n_total_leidos = n_leidos;
  if ((n_leidos) > 12)
    {
        while ((n_leidos) >0)
          {
              dato[0]=Serial.read();
                  if (dato[0]=='/')
          {
              for (i=0;i<=11;i++)
       {
           dato[i]=Serial.read();
      }
              for (i=0;i<6;i++)
       {
                              if (codigo_global[i]==dato[i])
                                {
                                    error=0;
                                }
            else
              {
          error=1;
          error_autenticacion=1;
          i=6;
              }
                          }
```

```
            for (i=0;i<3;i++)
                        {
                          codigo_orden[i]=dato[(i+6)];
                          codigo_orden_recibido[i]=codigo_orden[i];
                        }
            he_leido_datos=1;
            for (i=0;i<3;i++)
                        {
                          if(codigo_fin[i]==dato[(i+9)])
            {
                                error=0;
            }
                         else
            {
                                error=1;
                                error_fin_bloque=1;
                                codigo_orden[9]='_';
                                codigo_orden[10]='o';
                                codigo_orden[11]='k';
                                i=2;
            }
        }
            if (codigo_orden[0]=='S' && codigo_orden[1]=='i' && codigo_orden[2]=='1')
                        {
                          sinc=1;
                        }
// Borro datos sobrantes
            n_leidos=Serial.available();
            if (n_leidos>0)
              {
               for (i=0;i<n_leidos;i++)
                 {
                   basura[0]=Serial.read();
                 }
              }
              }
          n_leidos=Serial.available();
      }
      }
}
```

// RUTINA DE ESCRITURA POR EL PUERTO SERIE

```
//   Esta rutina realiza  la escritura por el puerto serie
controlador.                                                    //
//   Si está activada ("escribir=1") Construye el bloque de información con:
// 1.- El carácter de inicio de bloque ("/")
// 2.- El código global asignado a la pareja COORDINADOR-NODO
// 3.- El código de a orden mandada por el programa cíclico
// 4.- El código de fin de bloque ("_ok")
// 5.- Deja un registro de que ha realizado la escritura( he_escrito_datos )

void escribir_puertoserie()
{
        if (escribir==1)
                {
                        Serial.print('/');
                        for (i=0;i<6;i++)
                                {
                                        Serial.print(codigo_global[i]);
                                }
                        for (i=0;i<3;i++)
                                {
                                        Serial.print(codigo_orden[i]);
                                        codigo_orden_enviado[i]=codigo_orden[i];
                                }
                        for (i=0;i<3;i++)
                                {
                                        Serial.print(codigo_fin[i]);
                                }
                        he_escrito_datos=1;
                        escribir=0;
                        error_escribir=0;
                }
        else
                {
                        he_escrito_datos=0;
                        error_escribir=1;
                }
}
```

```
// RUTINA DE INTERPRETACIÓN DE ORDENES
//   Esta rutina realiza  la lectura del puerto serie del controlador.
// 1.- Comparo orden con el estado en el ciclo anterior
//   1.1.- Si es la misma, no hago nada
//   1.2.- Si es distinta, autorizo ejecución de orden
// 2.- Inicia temporizadores de intermitencia
// 3.- Predispone semáforos de intermitencia


void  interpretar_orden()
{
        If (codigo_orden_recibido[0]=='F' && codigo_orden_recibido[1]=='r' &&
codigo_orden_recibido[2]=='1' && freno_to==0)
                {
                        analogWrite(salidapwm[2],255);
                        freno=1;
                        cambio=4;
                }
 else
                {
                        If (codigo_orden[0]=='F' && codigo_orden[1]=='r' &&
                codigo_orden[2]=='0' && freno_to==1)
                                {
                                        analogWrite(salidapwm[2],0);
                                        freno=0;
                                        cambio=4;
                                }
 }
        If (((codigo_orden[0]=='D' && codigo_orden[1]=='e' ) || (codigo_orden[0]=='I'  &&
codigo_orden[1]=='z')) && emergencia_to==1)
                {
                        codigo_orden_recibido[0]=='E';
                        codigo_orden_recibido[1]=='m';
                        codigo_orden_recibido[2]=='1';
                }
        If (codigo_orden_recibido[0]=='E' && codigo_orden_recibido[1]=='m' &&
codigo_orden_recibido[2]=='1' && emergencia_to==0 )
                {
                        emergencia=1;
                        ciclo_emergencia=1;
                        izquierda=0;
                        ciclo_izquierda=0;
                        derecha=0;
                        ciclo_derecha=0;
                        emergencia_to=1;
                        cambio=3;
                        t_emer=(to+2000);
                }

         if (emergencia_to==0)
                {
                        If (codigo_orden_recibido[0]== 'D' && codigo_orden_recibido[1]
                == 'e'  && codigo_orden_recibido[2]=='1' && derecha_to==0)
     {
```

```
                                        derecha=1;
                                        derecha_to=1;
                                        izquierda=0;
                                        ciclo_derecha=1;
                                        ciclo_izquierda=0;
                                        ciclo_emergencia=0;
                                        cambio=2;
                                        t_de=(to+1000);
                                }
                        If (codigo_orden_recibido[0]=='I' &&
                codigo_orden_recibido[1]=='z' && codigo_orden_recibido[2]=='1' &&
                izquierda_to==0 )
                                {
                                        izquierda=1;
                                        izquierda_to=1;
                                        derecha=0;
                                        ciclo_derecha=0;
                                        ciclo_izquierda=1;
                                        ciclo_emergencia=0;
                                        cambio=1;
                                        t_iz=(to+1000);
                                }
                        If (codigo_orden_recibido[0]=='D' &&
                codigo_orden_recibido[1]=='e' && codigo_orden_recibido[2]=='0' &&
                derecha_to==1 )
                                {
        cambio=2;
        analogWrite(salidapwm[1],0);
        analogWrite(salidapwm[0],0);
        ciclo_emergencia=0;
        ciclo_derecha=0;
        ciclo_izquierda=0;
        t_emer=(to+2000);
                                }
                        If (codigo_orden_recibido[0]=='I' &&
                codigo_orden_recibido[1]=='z' && codigo_orden_recibido[2]=='0' &&
                izquierda_to==1 )
                                {
          cambio=1;
          analogWrite(salidapwm[0],0);
          analogWrite(salidapwm[1],0);
          ciclo_emergencia=0;
          ciclo_derecha=0;
          ciclo_izquierda=0;
          t_emer=(to+2000);
                                }

                }
         if(codigo_orden_recibido[0]=='E' && codigo_orden_recibido[1]=='m' &&
codigo_orden_recibido[2]=='0' && emergencia_to==1 )
                {
                        emergencia=0;
                        cambio=3;
                        analogWrite(salidapwm[1],0);
```

```
                        analogWrite(salidapwm[0],0);
                        ciclo_emergencia=0;
                        ciclo_derecha=0;
                        ciclo_izquierda=0;
                        t_emer=(to+2000);
                }
        freno_to=freno;
        emergencia_to=emergencia;
        derecha_to=derecha;
        izquierda_to=izquierda;
}


// RUTINA DE EJECUTAR ORDENES
//   Esta rutina se encarga de realizar la intermitencia de las luces
// 1.- miro el valor
//   1.1.- ciclo_emergencia en la señal de emergencia
//   1.2.- ciclo_derecha en la señal de derecha
//   1.3.- ciclo_izquierda en la señal de izquierda

void ejecutar_orden()
{
        if (ciclo_emergencia==1)
                {
                        if ((to < (t_emer - 1000)) && inter_em==0)
                                {
                                        analogWrite(salidapwm[1],255);
                                        analogWrite(salidapwm[0],255);
                                        inter_em=1;
                                }
                        if ( (to > (t_emer - 1000)) && (to < t_emer) && inter_em==1)
                                {
                                        analogWrite(salidapwm[1],0);
                                        analogWrite(salidapwm[0],0);
                                        inter_em=0;
                                }
                        if (to>t_emer)
                                {
                                        t_emer=(to+2000);
                                        inter_em=0;
                                }
                }

 if (ciclo_derecha==1)
                {
                        if ((to < (t_de - 500)) && inter_de==0)
                                {
                                        analogWrite(salidapwm[1],255);
                                        analogWrite(salidapwm[0],0);
                                        inter_de=1;
                                }
                        if ( (to > (t_de - 500)) && (to < t_de) && inter_de==1)
                                {
                                        analogWrite(salidapwm[1],0);
```

```
                                        analogWrite(salidapwm[0],0);
                                        inter_de=0;
                                }
                        If (to>t_de)
                                {
                                        t_de=(to+1000);
                                }
                }

  if (ciclo_izquierda==1)
                {
                        if ((to < (t_iz - 500)) && inter_iz==0)
                                {
                                        analogWrite(salidapwm[1],0);
                                        analogWrite(salidapwm[0],255);
                                        inter_iz=1;
                                }
                        if( (to > (t_iz - 500)) && (to < t_iz) && inter_iz==1)
                                {
                                        analogWrite(salidapwm[1],0);
                                        analogWrite(salidapwm[0],0);
                                        inter_iz=0;
                                }
                        if (to>t_iz)
                                {
                                        t_iz=(to+1000);
                                }
                }
}
```

10.2 CODIGO FUENTE NODO

// VARIABLES GLOBALES

```
// Para la configuración del controlador
int salidadig[6]={2,4,7,8,12,13};                                  // Array pines Digitales
int salidapwm[6]={3,5,6,9,10,11};                                  // Array pines PWM
int contenidoanalog[6];                                            // Vector entradas analógicas
char codigo_global[6]={'C','0','1','N','0','1'};                   // Código global
char codigo_fin[3]={'_','o','k'};                                  // Fin de información
char codigo_orden[3];                                              // Orden recibida/a transmitir
char codigo_orden_enviado[3];                                      // Orden enviada
char codigo_orden_recibido[3];                                     // Orden recibida

// para la rutina cíclica
int sinc;                          // Variable que indica si existe sincronizacion
int escribir;
int error_escribir;
int he_escrito_datos;
int i;
float to                                                           // Tiempo del ciclo
float to_ant                                                       // Tiempo del ciclo anterior
float t_fin;                                                       // Tiempo de fin de intermitencia
float t_sinc;                                                      // Tiempo de emisión sinc
float t_lat;                                                       // Temporizador para el latido
float temporizador_orden;          // Temporizador en la ejecución de una orden

// Para la rutina de lectura del puerto serie
int error;
int he_leido_datos;

// Para la rutina leo_entrada
int izquierda;                     // Estado del intermitente izquierdo en el ciclo
int derecha;                       // Estado  del intermitente derecho en el ciclo
int emergencia;                    // Estado de la emergencia en el ciclo
int freno;                         // Estado del freno en el ciclo
int bateria;                       // Estado de la batería en el ciclo
int izquierda_to;                  // Estado de la  izquierda en el ciclo anterior
int derecha_to;                    // Estado de la  derecha en el ciclo anterior
int emergencia_to;                 // Estado de la emergencia en el ciclo anterior
int freno_to;                      // Estado del freno en el ciclo anterior
int limite_bat;                    // Evitar rebotes en la alarma por batería baja
int cambio;                        // Cambio en los valores de las entradas
int cambio_fr;                     // Cambio en los valores de la entrada al freno
int cambio_em;                     // Alarma en la batería del nodo
int cambio_de;                     // Alarma en la batería del nodo
int cambio_iz;                     // Alarma en la batería del nodo
int bateria_to;                    // Estado de la batería en el ciclo anterior
int cambio_bat;                    // Alarma en la batería del nodo
int Led_bateria;                   //Semáforo de encendido de led de bateria
int Led_sinc;                      //Semáforo de encendido de led de sincronismo
int Led_freno;                     //Semáforo de encendido de led de freno
```

```
int Emergencia_ON;                  //Semáforo de encendido de led de emergencia
int Derecha_ON;                     //Semáforo de encendido de led Derecha
int Izquierda_ON;                   //Semáforo de encendido de led Izquierda
float inicio_cont;                  // Variables para generar un temporizador
float cont;                         // Temporizador en el ciclo actual
float cont_to;                      // Temporizador ene le ciclo anterior
float cont_em;                      // Temporizador de la emergencia
float cont_de;                      // Temporizador del intermitente derecho
float cont_iz;                      // Temporizador el intermitente izquierdo

// Para la rutina escribir_puertoserie
int inter_iz;                       // Actuación del intermitente izquierdo
int inter_de;                       // Actuación del intermitente derecho
int inter_em;                       // Actuación de la emergencia
int bateria_nodo;                   // Actuación en la batería

// Para la rutina escribir_actuadores
int iz_on;                          // Semáforo del intermitente izquierdo
int de_on;                          // Semáforo del intermitente derecho
int em_on;                          // Semáforo de la emergencia
int fr_on;                          // Semáforo del freno
```

// CONFIGURACIÓN SETUP

```
void setup()
{

//   LA CONFIGURACIÓN DEL CONTROLADOR NODO
//   A continuación se recoge el conjunto de instrucciones que permiten
//   configurar el Hardware del controlador NODO:
//   1.- Configurar el Puerto Serie

        for (i=0;i<6;i++)
                {
                        pinMode(salidadig[i],OUTPUT);
                        pinMode(salidapwm[i],OUTPUT);
                }
        Serial.begin(57600);

}

//        CONFIGURACIÓN INICIAL DE VARIABLES
//   A continuación se recojen el conjunto de variables que han de ser pre-
//   configuradas antes de iniciar el programa cíclico
//   1.- Sincronismo
//   2.- Temporizador de latido
//   3.- Temporizador de sobre tiempo

        for (i=0;i<6;i++)
                {
                        digitalWrite(salidadig[i],LOW);
                        analogWrite(salidapwm[i],0);
  }
  sinc=0;
  freno=0;
  emergencia=0;
  derecha=0;
  izquierda=0;
  freno_to=0;
  emergencia_to=0;
  derecha_to=0;
  izquierda_to=0;
 ciclo_emergencia=0;
  ciclo_derecha=0;
  ciclo_izquierda=0;
  limite_bat=650;
  t_lat=(millis()/1000)+60;
  to=millis();
  to_ant=millis();
 t_fin=(millis()/1000)+40;
  t_sinc=(millis()/1000)+10;
}
```

// PROGRAMA CICLICO

```
void loop()
{
//   ACCIONES  AL COMENZAR UN CICLO DE PROGRAMA
//   Antes de comenzar cada ciclo de programa haya que realizar una serie de
//   acciones:
//   1.- iniciar las variables globales
//   2.- Definir las variables que se utilizarán en la Rutina
//   3.- Leer el tiempo en la que se ejecuta el ciclo
//   4.- Corregir errores en lectura de tiempo

error=0;
error_escribir=0;
to=millis();
if (to < to_ant)
        {
                t_lat=(to/1000)+40;
                t_fin=(to/1000)+20;
                t_sinc=(to/1000)+10;
                cont_de=(to/1000)+2;
                cont_iz=(to/1000)+2;
        }
  else
  {
                to_ant=to;
  }

// El NODO ha de sincronizarse con el COORDINADOR
// 1.- Emite el bloque de sincronismo cada 10 segundos
// 2.- Si recibe respuesta se sincroniza y enciende el diodo de sincronismo

if(sinc==0)
        {
                if (to/1000 > t_sinc)
                        {
                                codigo_orden[0]='S'
              codigo_orden[1]='i';
              codigo_orden[2]='1';
              escribir=1;
              escribir_puertoserie();
                                t_sinc=(to/1000)+10;
             }
            leer_puertoserie();
            if(he_leido_datos >0 && error==0 && sinc==1)
             {
               escribir_salidas();
             }
        }
```

```
//   Si está sincronicado, el COORDINADOR ha de realizar cíclicamente:
//   1.- Sincronizarse y garantizar el sincronismo (latido)
//   2.- Leer las órdenes que llegan por las entradas analógicas del NODO
//   3.- Si hay cambios, Escribir por el puerto serie la orden al COORDINADOR
//   4.- Recibir la contestación del COORDINADOR
//   5.- Encender diodos en el NODO.

 if (sinc==1)
         {
                  leer_entradas();
     if (cambio>0)
       {
         escribir_actuadores();
                           }
     leer_puertoserie();
    if(he_leido_datos >0)
       {
                  escribir_salidas();
         }
}
```

// RUTINA PARA LA LECTURA DE ENTRADAS

```
void leer_entradas()
{
  // 1.- Defino variables locales
  // 2.- Leo las entradas analógicas del controlador
  // 3.- Genero el estado actual y comparo cambios

  int lectura_A0;
  int lectura_A1;
  int lectura_A2;
  int lectura_A3;
  cambio=0;
  cambio_fr=0;                                                    // cambio en freno
  cambio_em=0;                                                  // cambio en emergencia
  cambio_de=0;                                                    // cambio en derecha
  cambio_iz=0;                                                   // cambio en izquierda
  cambio_bat=0;                                                   // cambio en bateria
    lectura_A0=analogRead(A0);
    lectura_A1=analogRead(A1);
    lectura_A2=analogRead(A2);
    lectura_A3=analogRead(A3);
    delay(1);
  for(i=0;i<3;i++)                                  // promedio las lecturas de las entradas
 {
                    lectura_A0=(analogRead(A0)+lectura_A0);
                    lectura_A1=(analogRead(A1)+lectura_A1);
                    lectura_A2=(analogRead(A2)+lectura_A2);
                    lectura_A3=(analogRead(A3)+lectura_A3);
                    delay (1);
         }
  lectura_A0=lectura_A0/4;
  lectura_A1=lectura_A1/4;
```

```
lectura_A2=lectura_A2/4;
lectura_A3=lectura_A3/4;

// LECTURA DEL FRENO
//    El freno actúa siempre que se tenga pulsado el sensor del freno
//    1.- Es independiente del estado de cualquier otro sensor
//    2.- No lleva ninguna condición el tiempo de encendido
//    3.- El COORDINADOR informa sobre su encendido y apagado

 if (lectura_A2 <=75 && freno_to==0)
        {
                freno=1;                // El freno se activa poniéndolo a tierra
                cambio_fr=1;
        }
 if (lectura_A2 >=950 && freno_to==1)
        {
                freno=0;
                cambio_fr=1;
        }
 if (lectura_A2 >75 && lectura_A2 <950)
        {
                cambio_fr=0;
        }

// LECTURA DE LA EMERGENCIA
//    La Emergencia actúa siempre que se tenga ACTUADO el interruptor
//    1.- Inactiva los intermitentes Izquierdo y Derecho
//    2.- No lleva ninguna condición el tiempo de encendido

 if (lectura_A1 >=950 )
        {
          if(emergencia_to==0)
                 {
                        cambio_em=1;
                        emergencia=1;       //  La emergencia se activa a 5V.
                        izquierda=0;
                        derecha=0;
                 }
          else
                {
                       cambio_em=0;
                }
        }
 if (lectura_A1 <950)
        {
                if(emergencia_to==1)
                      {
                             cambio_em=1;
                             emergencia=0;
                             izquierda=0;
                             derecha=0;
                      }
                else
```

```
                        {
                                cambio_em=0;
                        }
        }

// LECTURA DEL INTERMITENTE DERECHO
//   El intermitente DERECHO es actuado mediante la acción sobre el joystick
//   1.- El tiempo de encendido mínimo es de 2 sg.
//   2.- Si no se apaga manualmente, a los 20sg se apaga automáticamente
//   3.- El COORDINADOR informa sobre su encendido y apagado
  if (lectura_A0 >=950 && emergencia==0 && (to/1000)+2>cont_de)
        {
          cambio_de=1;
          cont_de=(to/1000)+2;
          if(derecha_to==1)
          {
            derecha=0;
          }
          else
          {
            derecha=1;
            izquierda=0;
            cambio_de=0;
          }
        }

// LECTURA DEL INTERMITENTE IZQUIERDO
//   El intermitente IZQUIERDO es actuado mediante el  joystick
//   1.- El tiempo de encendido mínimo es de 2 sg
//   2.- Si no se apaga manualmente, a los 20sg se apaga automáticamente
//   3.- El COORDINADOR informa sobre su encendido y apagado
  if (lectura_A0 <=75 && to/1000>cont_iz && emergencia_to==0)
          {
                  if(izquierda_to==0)
                          {
                                  cambio_iz=1;
                                  izquierda=1;
                                  derecha=0;
                                  cont_iz=(to/1000)+2;
                                  izquierda_to=1;
                                  derecha_to=1;
                          }
           else
                          {
                                  cambio_iz=1;
                                  izquierda=0;
                                  derecha=0;
                                  cont_iz=(to/1000)+2;
                          }
          }
  else
        {
```

```
                cambio_iz=0;
        }

// LECTURA DEL ESTADO DE LA BATERIA
//  La alarma de la BATERÍA es generada sensando su tensión mediante
//  un divisor resistivo.
//  1.- Cuando la alarma está actuada se eleva la tensión límite
//      para evitar rebotes
//  2.- Estará encendida mientra no se recargue la batería
//  3.- El COORDINADOR no informa sobre su batería

 if (lectura_A3 <=limite_bat)
        {
                if (bateria_to==0)
                        {
                                cambio_bat=1;
                                bateria=1;
                                limite_bat=700;
                                bateria_to=1;
                        }
                else
                        {
                                cambio_bat=0;
                        }
        }
 else
        {
                if (bateria_to==1)
                        {
                                cambio_bat=5;
                                bateria=0;
                                limite_bat=650;
                                bateria_to=0;
                        }
                else
                        {
                                cambio_bat=0;
                        }
        }

 if(cambio_bat>0)
        {
                cambio=5;
        }
 else
  {
                if(cambio_fr>0)
     {
       cambio=4;
     }
                else
     {
       if(cambio_em>0)
        {
```

```
            cambio=3;
          }
        else
          {
            if(cambio_de>0)
              {
                cambio=2;
              }
              else
               {
                 if (cambio_iz>0)
                   {
                     cambio=1;
                   }
               }
          }
      }
  }
  if( cambio_bat==0 && cambio_fr==0 && cambio_em==0 && cambio_de==0 &&
cambio_iz==0)
          {
                  cambio=0;
          }
    freno_to=freno;
    emergencia_to=emergencia;
    derecha_to=derecha;
    izquierda_to=izquierda;
    bateria_to=bateria;
}

// RUTINA CONFIGURAR PARA ESCRIBIR EN PUERTO SERIE
// Esta rutina se necarga de;
// 1.- Predisponer los valores correctos de la orden a transmitir

void  escribir_actuadores()
{
   if(cambio)
  {

    switch(cambio)
    {
          case 4:
                  codigo_orden[0]='F';
                  codigo_orden[1]='r';
                  codigo_orden[2]=freno + '0';
                  break;


          case 3:
                  codigo_orden[0]='E';
                  codigo_orden[1]='m';
                  codigo_orden[2]=emergencia + '0';
                  break;
```

```
        case 2:
                codigo_orden[0]='D';
                codigo_orden[1]='e';
                codigo_orden[2]=derecha + '0';
                break;

        case 1:
                codigo_orden[0]='I';
                codigo_orden[1]='z';
                codigo_orden[2]=izquierda + '0';
                break;

    }
    escribir=1;
    escribir_puertoserie();
  }
}
```

// RUTINA DE ESCRITURA POR EL PUERTO SERIE

```
//   Esta rutina realiza  la escritura por el puerto serie
controlador.                                                    //
//   Si está activada ("escribir=1") Construye el bloque de información con:
// 1.- El carácter de inicio de bloque ("/")
// 2.- El código global asignado a la pareja COORDINADOR-NODO
// 3.- El código de a orden mandada por el programa cíclico
// 4.- El código de fin de bloque ("_ok")
// 5.- Deja un registro de que ha realizado la escritura( he_escrito_datos )

void escribir_puertoserie()
{
        if (escribir==1)
                {
                        Serial.print('/');
                        for (i=0;i<6;i++)
                                {
                                        Serial.print(codigo_global[i]);
                                }
                        for (i=0;i<3;i++)
                                {
                                        Serial.print(codigo_orden[i]);
                                        codigo_orden_enviado[i]=codigo_orden[i];
                                }
                        for (i=0;i<3;i++)
                                {
                                        Serial.print(codigo_fin[i]);
                                }
                        he_escrito_datos=1;
                        escribir=0;
                        error_escribir=0;
                }
        else
                {
                        he_escrito_datos=0;
```

```
                    error_escribir=1;
                }
}
```

// RUTINA DE LECTURA DEL PUERTO SERIE

```
//   Esta rutina realiza la lectura del puerto serie del NODO//
// 1.- Defino variables locales
// 2.- Inicializo variables
// 3.- Lee el bloque de información recibido
//   3.1.- Longitud fija de 13 caracteres
//   3.2.- Detecta el inicio de bloque de información
//   3.3.- Autentica con código global asignado a COORDINADOR-NODO
//   3.4.- Extrae la orden recibida
//   3.5.- Comprueba que es una orden válida
// 4.- Detecta la señal de sincronismo
// 5.- Detecta la señal de latido
// 6.- Deja registros de lectura de datos y de errores

void leer_puertoserie()
{
  int n_total_leidos;                          // El  número de caracteres recibidos
        int n_leidos=0;                        // El número de caracteres por leer
        int error_autenticacion=0;             // Error en la autenticación del bloque
  int error_fin_bloque=0;                      // El fin de bloque no es correcto
 char dato[12];                                // variable con caracteres recibidos válidos
  char basura[1];                             // variable para borrar datos no válidos
  error=0;
  he_leido_datos=0;
  for (i=0;i<12;i++)
    {
        dato[i]=' ';
    }
  n_leidos=Serial.available();
  n_total_leidos = n_leidos;
  if ((n_leidos) > 12)
    {
        while ((n_leidos) >0)
          {
              dato[0]=Serial.read();
                  if (dato[0]=='/')
          {
              for (i=0;i<=11;i++)
       {
           dato[i]=Serial.read();
      }
              for (i=0;i<6;i++)
       {
                              if (codigo_global[i]==dato[i])
                {
                    error=0;
                }
            else
              {
          error=1;
```

```
            error_autenticacion=1;
            i=6;
                }
                            }
                for (i=0;i<3;i++)
                            {
                              codigo_orden[i]=dato[(i+6)];
                              codigo_orden_recibido[i]=codigo_orden[i];
                            }
                he_leido_datos=1;
                for (i=0;i<3;i++)
                            {
                              if(codigo_fin[i]==dato[(i+9)])
                {
                                    error=0;
                }
                             else
                {
                                    error=1;
                                    error_fin_bloque=1;
                                    codigo_orden[9]='_';
                                    codigo_orden[10]='o';
                                    codigo_orden[11]='k';
                                    i=2;
                }
            }
                if (codigo_orden[0]=='S' && codigo_orden[1]=='i' && codigo_orden[2]=='1')
                            {
                              sinc=1;
                            }
                if (codigo_orden[0]=='L' && codigo_orden[1]=='a' && codigo_orden[2]=='t')
                            {
                              t_lat=(to/1000)+120;
                              sinc=1;
                            }
// Borro datos sobrantes
                n_leidos=Serial.available();
                if (n_leidos>0)
                  {
                   for (i=0;i<n_leidos;i++)
                     {
                       basura[0]=Serial.read();
                     }
                  }
                  }
             n_leidos=Serial.available();
        }
        }
}
```

```
// RUTINA DE ESCRITURA DE LAS SALIDAS EN EL NODO
// Esta rutina se encarga de
// 1.- Encender el led de sincronismo
// 2.- Encender los led de intermitentes
// 3.- Encender el led de batería baja del nodo

void escribir_salidas()
 {
        if(codigo_orden_recibido[0]=='S' && codigo_orden_recibido[1]=='i')
  {
    if(codigo_orden_recibido[2]=='1' && Led_sinc==0)
     {
                                        Led_sinc=1;
                                        digitalWrite(4,HIGH);
                                }
    if(codigo_orden_recibido[2]=='0' && Led_sinc==1)
     {
                                        Led_sinc=0;
                                        digitalWrite(4,LOW);
                                }
                }
        if(codigo_orden_recibido[0]=='F' && codigo_orden_recibido[1]=='r')
                {
                        if(codigo_orden_recibido[2]=='1' && Led_freno==0 )
                                {
                                        Led_freno=1;
                                        digitalWrite(13,HIGH);
                                }
                        if(codigo_orden_recibido[2]=='1' && Led_freno==1 )
                                {
                                        Led_freno=0;
                                        digitalWrite(13,LOW);
                                }
                }
        if(codigo_orden_recibido[0]=='E' && codigo_orden_recibido[1]=='m')
                {
                        if( codigo_orden_recibido[2]=='1' && Emergencia_ON==0)
                                {
                                        Emergencia_ON=1;
                                        digitalWrite(7,HIGH);
                                        digitalWrite(8,HIGH);
                                }
                        if( codigo_orden_recibido[2]=='1' && Emergencia_ON==1)
                                {
                                        digitalWrite(7,LOW);
                                        digitalWrite(8,LOW);
                                }
                }
         if(codigo_orden_recibido[0]=='D' && codigo_orden_recibido[1]=='e')
                {
                        if( codigo_orden_recibido[2]=='1' && Derecha_ON==0)
                                {
                                        Derecha_ON=1;
                                        digitalWrite(7,HIGH);
```

```
                                        digitalWrite(8,LOW);
                                }
                        if( codigo_orden_recibido[2]=='0' && Derecha_ON==1)
                                {
                                        digitalWrite(7,LOW);
                                        digitalWrite(8,LOW);
                                }
                }
          if(codigo_orden_recibido[0]=='l' && codigo_orden_recibido[1]=='z')
        {
          if( codigo_orden_recibido[2]=='1' && Izquierda_ON==0)
              {
                                        Izquierda_ON=1;
                                        digitalWrite(7,LOW);
                                        digitalWrite(8,HIGH);
                                }
          if( codigo_orden_recibido[2]=='0 && Izquierda_ON==1)
              {
                                                digitalWrite(7,LOW);
                                        digitalWrite(8,LOW);
                                }
                }

// Analizamos el estado de la Batería
          if (bateria_nodo==0 && Led_Bateria==0)
                {
                        Led_Bateria=1;
                        digitalWrite(12,HIGH);
                }
          if (bateria_nodo==1 && Led_Bateria==1)
                {
                        Led_Bateria=0;
                        digitalWrite(12,LOW);
      }
}
```